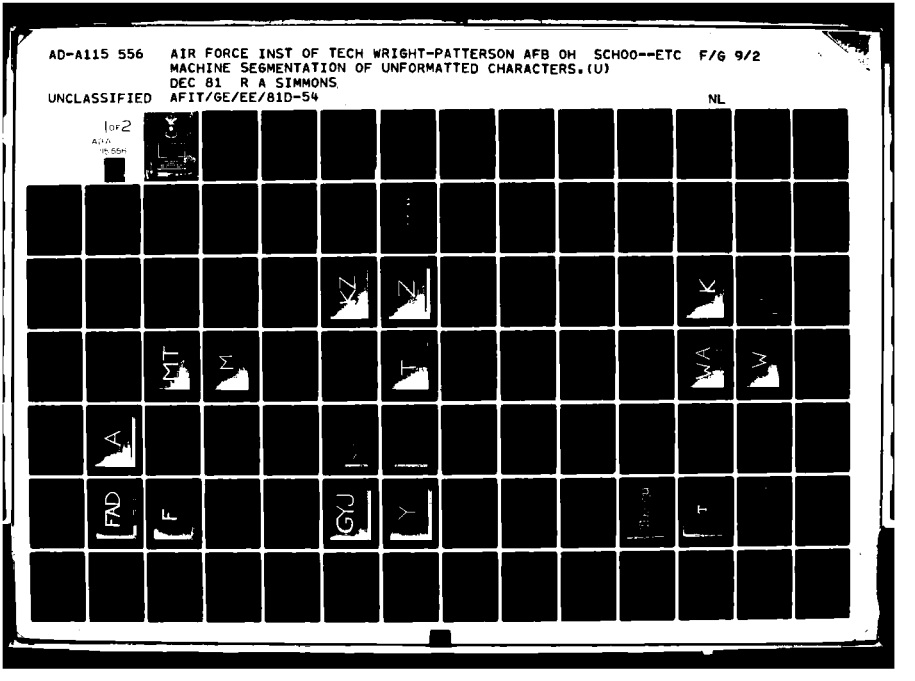
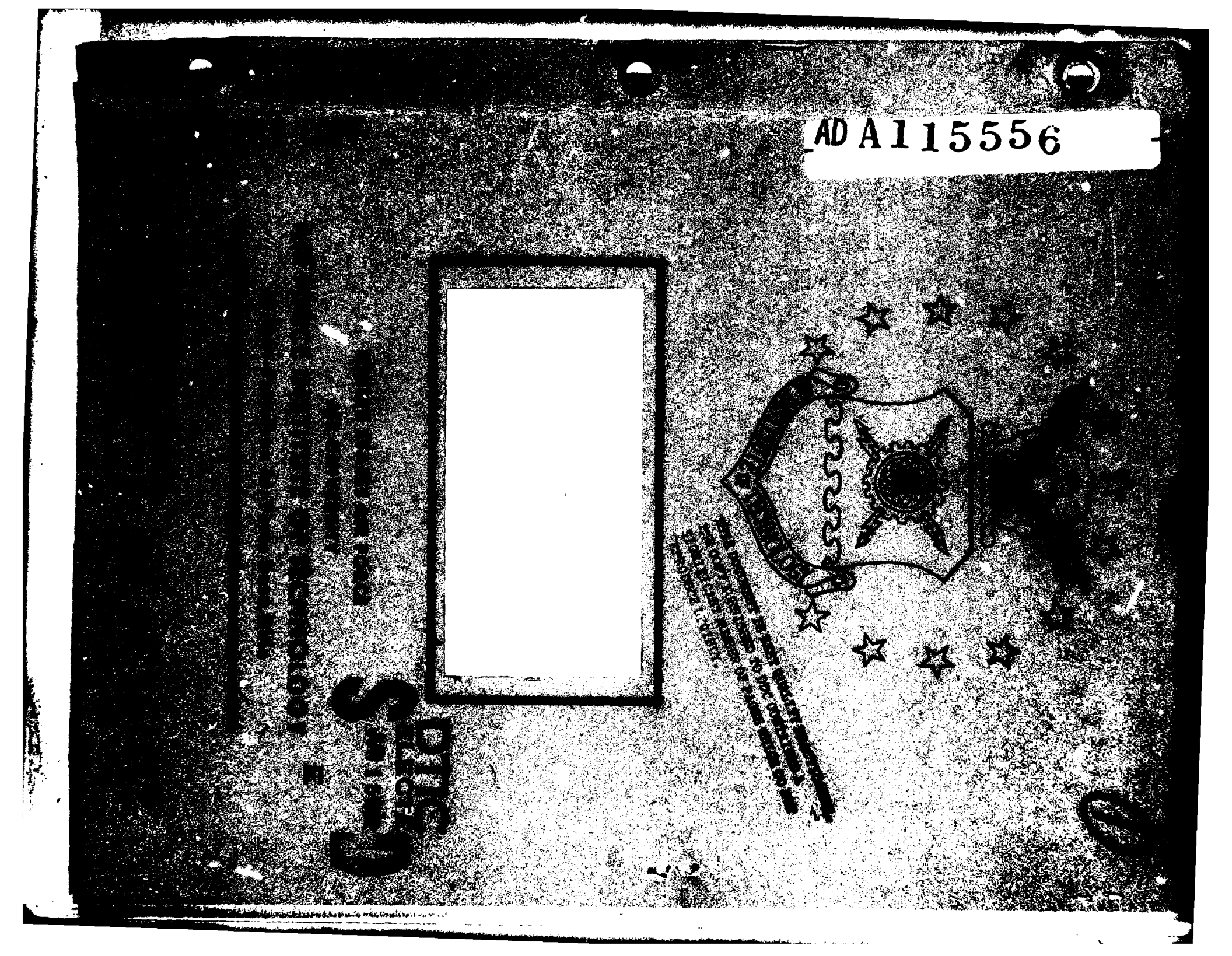
AD-A115 556 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOO--ETC F/G 9/2
MACHINE SEGMENTATION OF UNFORMATTED CHARACTERS.(U)
DEC 81 R A SIMMONS
UNCLASSIFIED AFIT/GE/EE/81D-54 NL





DISCLAIMER NOTICE

THIS DOCUMENT IS BEST QUALITY PRACTICABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

AFIT/GE/EE/81D-54

(1)

MACHINE SEGMENTATION OF UNFORMATTED CHARACTERS

THESIS

AFIT/GE/EE/81D-54 Robin Simmons
2Lt USAF

AFIT/GE/EE/81D-54

MACHINE SEGMENTATION OF UNFORMATTED CHARACTERS

THESIS

Presented to the Faculty of the School of Engineering
Of the Air Force Institute of Technology
Air University
in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

by

Robin A. Simmons, B.S.E.E.
Second Lieutenant USAF

Graduate Electrical Engineering

December 1981

Approved for public release; distribution unlimited.

Preface

While working on this thesis, I received assistance from many people. My thesis advisor, Dr. Matthew Kabrisky, was most helpful; he offered many suggestions along the way. I also must thank my readers: Captain Larry Kizer for sharing his knowledge and resources concerning the Data General Computer System, and Major Alan Ross, who suggested many excellent ideas in our early discussions and when reviewing this thesis.

Most importantly, I would like to thank my wife for her support while I worked on this thesis.

Robin A. Simmons

Contents

List of Figures

Figures (cont.)

List of Tables

Abstract

This thesis presents a method of segmenting unformatted alpha-numeric characters. Reconstructing the magnitude of the Fourier transform of a template character with the phase of a string of unformatted characters containing the template character causes all characters that do not have the magnitude of the template to be attenuated in the visual domain. The template character will not be attenuated, since it has both proper magnitude and phase, and a peak detector can find the most probable location of the character. This process also gives a first choice for what the character is (the template). Results are presented for most of the English alphabet.

I. Introduction

General.

Many U. S. Government agencies process thousands of pages of printed material each month. To store this information in easily retrievable and processable form, it has to be entered into a computer. Presently, typists perform this task, but if a machine could read from these sources without human intervention, then the input process could go at machine speeds, which are much faster. This could potentially save a large amount of human labor.

Background.

All current Optical Character Recognition (OCR) techniques require prior knowledge of the location and form of each character. Many useful OCR methods are available, and they work reasonably well if the characters are typewritten or carefully handwritten in pre-printed boxes [1-4]. Most existing text is not so formatted; some examples are:

a. Books
b. Journals
c. Magazines
d. Other published material
e. ZIP Codes or complete addresses

"Segmentation" is the term used to describe the process of separating the characters from each other in a line of print. Most people have seen order forms that can be machine-read if characters are carefully placed in pre-printed boxes. (See figure 1.1.) This is one form of segmentation. But this

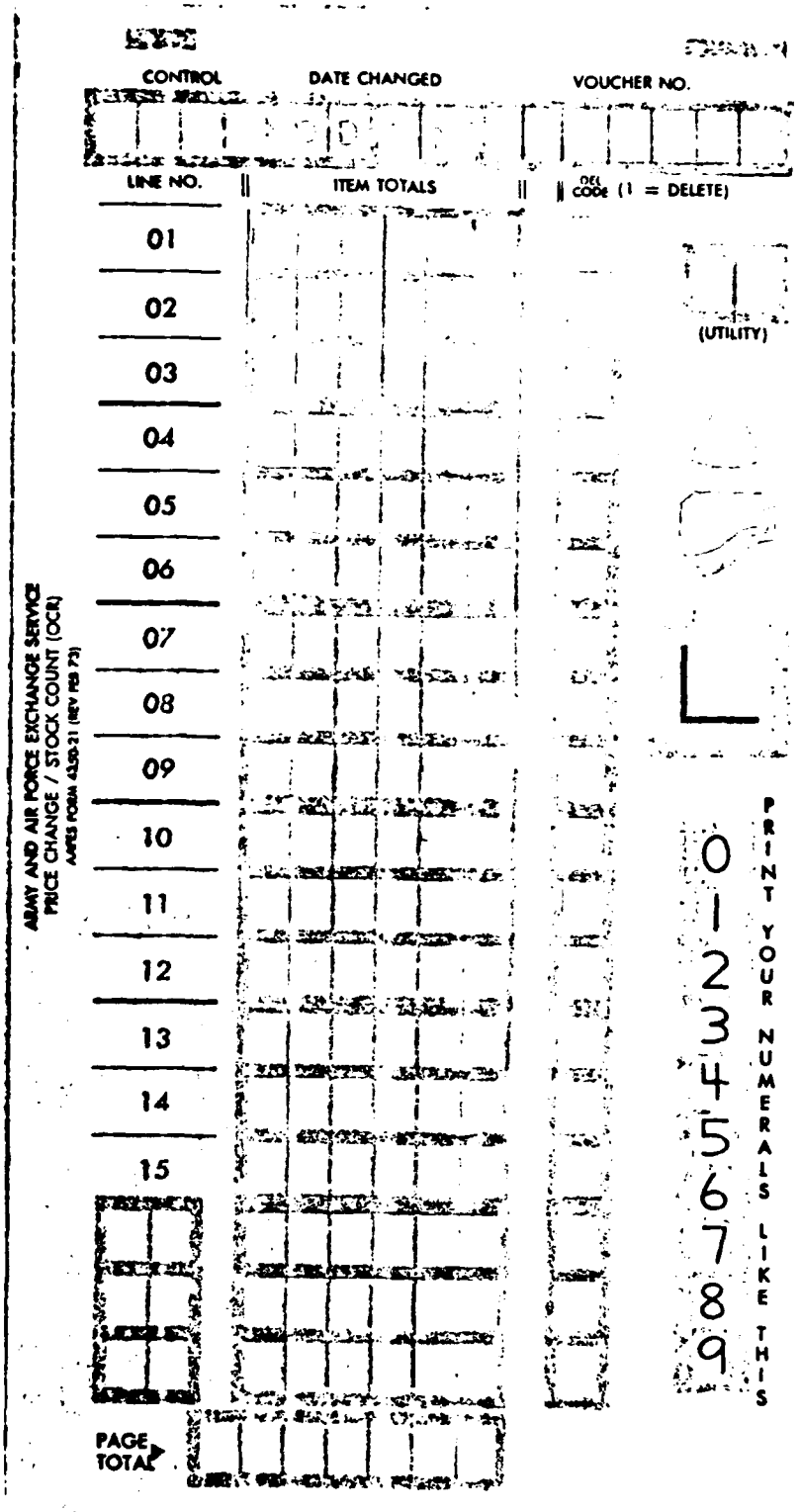
type of segmentation is not available when machine-reading from the examples listed above. An OCR technique that did not require the input characters to be specifically formatted for OCR purposes would be much more useful.

Problem.

There were had two research goals. The first was to develop software to perform Horev's algorithm [5] on the Data General Corporation computers in the Signal Processing Laboratory. This goal was accomplished. The second was to test the feasibility of this approach for segmenting unformatted alpha-numeric characters. This report contains the results of the feasibility study.

Scope.

This thesis project progressed through three stages: (1) background research, (2) implementation, and (3) analysis of results. The background research was limited to studying the target-location method developed by Moshe Horev. His Master's thesis [5] desribes a method of finding a target in a cluttered scene. Segmentation of characters can be thought of as finding a target (a character location) in a cluttered scene (the line of print). During implementation, the study was limited to English alpha-numeric characters, since results should be extendable to all shapes of characters and numbers. Hand-drawn characters were often used so that no template would exactly match the character in the input scene, and



Sample OCR Form with Segmentation Boxes
Figure 1.1

so that the characters could be made different sizes to study a size effect that is discussed in the next chapter. The results are presented visually in this report to clearly show the results and simplify the analysis.

Assumptions.

A character is segmented if it is bounded from left to right. It is assumed that a threshold program can put a "window" around the segmented character if its left-most and right-most boundaries are not removed by the process. Also, the height of the characters are assumed to be set to a given height (determined by a line-location process) before this segmentation algorithm is used [2:1530].

Overview of Chapters.

The next chapter contains an explanation of the theory behind Horev's algorithm, with emphasis on the two-dimensional discrete Fourier transform (2D-DFT). The third chapter contains a general description of the software development. Examples and an interpretation of the results are given in chapter four. Finally, chapter five gives my conclusion and recommendations.

II. Reconstruction of the Fourier Magnitude of One Scene with the Fourier Phase of Another

General.

To process any visual information, a domain has to be chosen that supports a useful calculation. Some researchers work solely in the spatial domain--the computations are performed on numerical values of brightness at discrete points across the two-dimensional picture. These points are called pixels (from "picture elements"). Other researchers [6,7] have suggested that the human visual system may perform its calculations in the spatial frequency domain, and logically, the greatest amount of success would occur if machines also processed in the frequency domain. The discrete Fourier transform (DFT) is used to compute the real and imaginary frequency components of the pixels. AFIT student Moshe Horev designed an algorithm to locate targets in a cluttered scene, performing his calculations in the frequency domain [5:9]. This thesis applies a much-simplified version of his method to the problem of character segmentation. Although the computation occurs in the frequency domain, the results will be presented in the visual (spatial) domain.

In two-dimensional functions (such as pictures), the phase of the signal contains most of the form information. The importance of this will be discussed at the end of this

chapter; first, a general description of the mathematics used in this segmentation algorithm will be given.

Two-Dimensional Fourier Transform.

To transform the discrete pixels from the visual domain to the frequency domain, a two-dimensional discrete Fourier transform (2D-DFT) is used. The pertinent equation for the 2D-DFT is [8:117]:

$$F(x,y) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n) \exp[-j(\frac{2\pi xm}{M})]\exp[-j(\frac{2\pi yn}{N})] \quad 2.1$$

between $0 \leq n \leq N-1$ and $0 \leq m \leq M-1$. Here, f(m,n) is the brightness value of the pixel at location (m,n) where m and n are summed over the area, and F(x,y) is the Fourier transform at any location (x,y).

This summation can be simplified if the dimensions N and M are equal and are powers of two, since a recursive approach can be efficiently used to perform the Fourier transform. Specifically, the algorithm I used [written by Ronald W. Schaefer, 30 June 1978] operates only on a square array with dimensions which are powers of two. It performs N one-dimensional Fourier transformations on the N rows of the two-dimensional array, then transposes the array, and finishes by again performing N one-dimensional Fourier transforms. The result is a 2D-DFT [8:320-321] that is transposed.

Similarly, the inverse two-dimensional Fourier transform (2D-IDFT) is computed by this equation:

$$f(m,n) = \frac{1}{XY} \sum_{x=0}^{X-1} \sum_{y=0}^{Y-1} F(x,y) \exp[j(\frac{2\pi mx}{X})]\exp[j(\frac{2\pi ny}{Y})] \quad 2.2$$

between $0 \leq x \leq X-1$ and $0 \leq y \leq Y-1$.

Computation of Magnitude and Phase.

The magnitude of the complex Fourier transform was calculated by the standard equation (eqn. 2.3):

$$MAG_i = [REAL_i^2 + IMAG_i^2]^{1/2} \quad 2.3$$

Similarly, the phase was calculated by equation (2.4):

$$ANG_i = \tan^{-1} [\frac{IMAG_i}{REAL_i}] \quad 2.4$$

except when $MAG_i = 0$, in which case $ANG_i = 0$.

Reconstruction of the Template Magnitude with the Phase of a Scene.

From Horev's work, the key to segmentation is reconstructing the magnitude of a template with the phase of a scene; this creates a new complex scene where segmentation is possible by a threshold algorithm. The real component of the new complex number is (equation 2.5):

$$REAL_i = ENM_i * \cos (ANG_i) \quad 2.5$$

and the imaginary component is (equation 2.6):

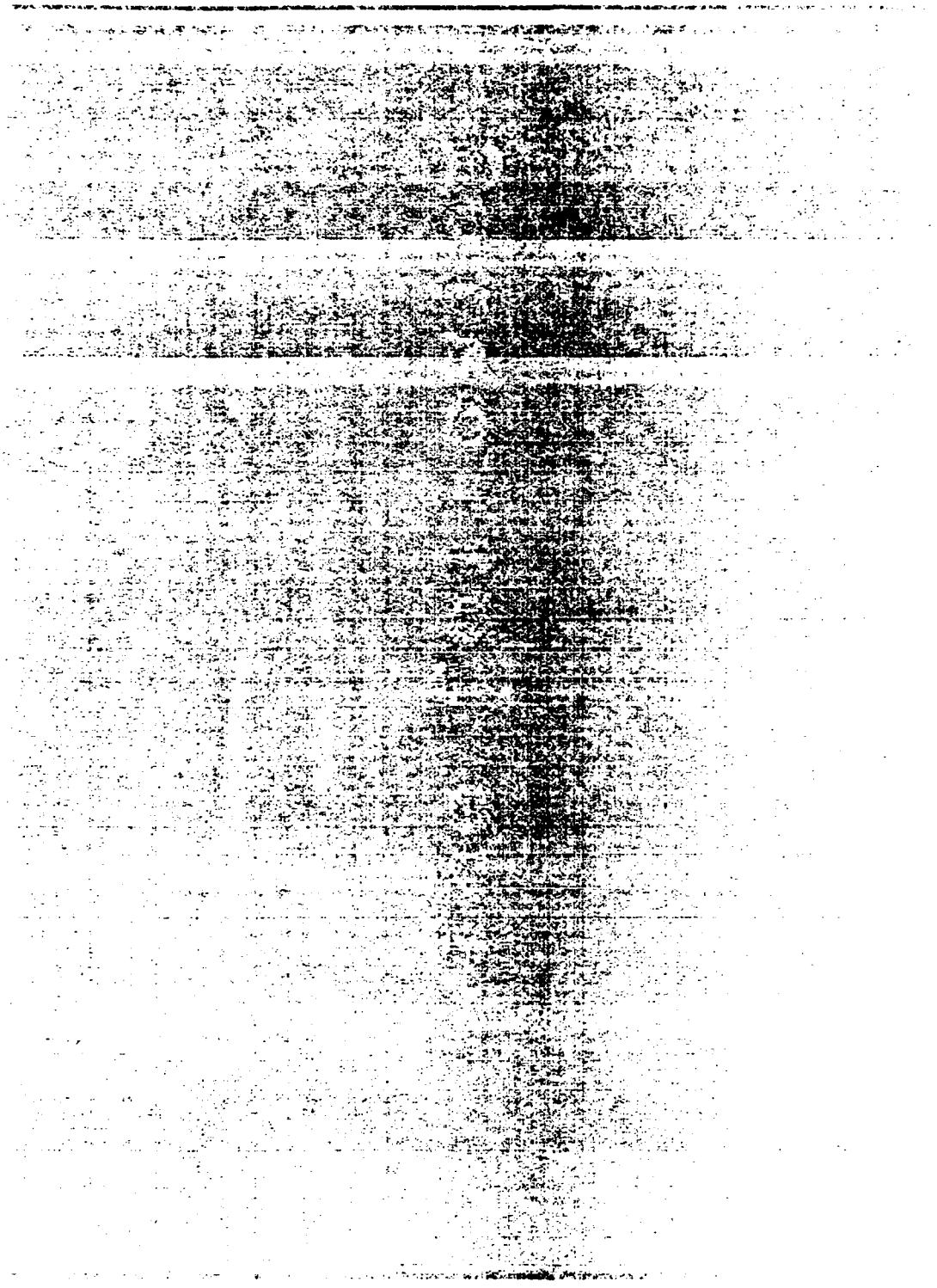
$$IMAG_i = ENM_i * \cos (ANG_i) \quad 2.6$$

where ENM_i is the energy-normalized value of MAG_i. The normalization factor is the square root of the sum of the squares of each magnitude component. By repeated application of equations 2.5 and 2.6, a complex file is recreated from a magnitude file and a phase file. Note that if the magnitude and phase of the same scene were recombined in this manner, the original (complex) scene would be recreated, but the pixels would be energy-normalized.

Importance of Phase in a Scene.

Research has shown that most of the form information in a scene is contained in the phase of its frequency components [9]. For a typical scene, a recognizeable version of the scene is reproduced if a file of unity magnitudes is reconstructed with the phase of the scene (and then inverse Fourier-transformed; see figure 2.1). Having the whole scene present in the reconstructed scene will not aid segmentation; the reconstructed scene will practically match the original scene. However, the size of a character relative to the 2D-DFT window size changes the amount of this effect.

Segmentation can be accomplished if the characters in the 2D-DFT window are relatively large. Normally, the most energy in a frequency spectrum is clustered around the DC term, that is, the energy is in the low frequency components. When unity magnitudes are reconstructed with the phase of a scene, the higher frequencies tend to be emphasized. This will be seen visually as the edges in a scene. When a larger character is in the window, it contains mostly low-frequency



Scene Reconstructed from Phase with Unity Magnitude
Figure 2.1

components (each stroke in the character will have more pixels to represent it and they will be nearly the same level of black; thus there will be few high frequency changes). The low-frequency components are not emphasized when reconstructed with unity magnitudes, and this scene will not be recognizeable. For the character to be visible, it will also need its magnitude components. Thus, segmentation is more likely to be possible if the character size-to-window size ratio is large enough to keep the phase from accurately representing the scene. Furthermore, in the actual reconstruction process, the magnitude file has most of its energy in the lower frequency components. This will cause the high frequency components of the target character to be somewhat de-emphasized back to their normal levels.

In this research, recognizeable characters were still present in phase-only scenes when one of the characters already filled 75% of the window (in height; i. e., about 50% of the window area). To simplify this work, only large characters (total character-to-window ratio above 75%) were used, except when working with original characters that were too small to optically enlarge to that ratio. The next chapter describes all of the equipment and software used to perform this reconstruction algorithm.

III. Software Development

General.

A Signal Processing Laboratory (SPL) has been developed in the Electrical Engineering department at AFIT. To most effectively use this resource, software must be developed to process digitized signals (audio and video, although this thesis includes only video). As a foundation for further video-processing software, this chapter contains three sections: the first section is a general description of the hardware available in the SPL at the time of this writing. The second describes the necessary details of digitization and processing so that the reader has a grasp of what is necessary to write software to process video. Finally, the third section will briefly describe the video-processing software that was developed concurrently in the SPL. Source listings of the programs are given in the Appendix.

Hardware.

The first step in understanding how video information is processed is to gain a working knowledge of the equipment currently available in the SPL. The principal components of the system are two minicomputers and a microcomputer. They are:

a. Data General Nova 2 processor
b. Data General Eclipse S/250 processor
c. Cromemco Z-2D microcomputer

The two Data General machines are used for the actual data processing, while the Cromemco computer is a Z-80 based processor that controls the analog-to-digital and digital-to-analog conversion devices. The A/D and D/A boards were built by Tecmar, Inc. and they have the capability of digitizing pictures of 64, 128, 256, or 512 pixels per line by 64, 128, or 256 lines. Input is from a standard studio vidicon camera (Cohu Electronics, Inc. model 6150-000), and output is to any standard monitor (or television set, if the signal is modulated onto a standard carrier frequency).

The Nova computer starts and stops the Cromemco via a fortran subroutine call named CHANNEL [10]. This subroutine performs two tasks; one is to pass proper parameters to the Cromemco for signal handling, and the other is to retrieve or send the actual data. The Z-2D is an eight bit processor, while both Data General machines have sixteen bit words. The subroutine call performs the necessary conversion to interface the machines and writes or reads directly to the disk drives. (The SPL currently has a ten megabyte and two twenty megabyte Data General disk drives.) Once the digitized picture is stored on disk, either the Nova or the Eclipse can access the information and process it. However, the Eclipse cannot directly communicate with the Cromemco; it can only access the disk drives on which the Nova stores the video data.

The steps necessary to handle video signal processing are:

Input: a. Activate vidicon, Cromemco, and Nova systems
b. Call subroutine CHANNEL from Nova to operate A/D and transfer data from Cromemco to Nova
c. Write video data to disk

Process data: a. Nova
b. Eclipse

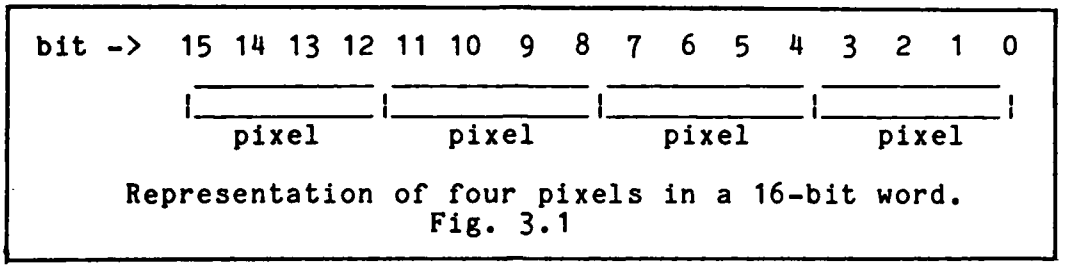
Output: a. Insure that video data is on a disk that is accesible to the Nova
b. Activate Cromemco, Nova, and output device (e. g. monitor)
c. Call subroutine CHANNEL to transfer data from Nova to Cromemco for D/A

Note that the Cromemco computer is completely transparent to the Nova; the user simply calls the subroutine and the Nova automatically either inputs the video frame that the Cromemco has ready or outputs a video frame to the Cromemco.

Video Digitization and Processing.

The software running in the Cromemco and switches on the video digitization boards set certain constraints on the form of the video data. The picture frame is usually digitized at 256 by 256 pixels. Furthermore, the Tecmar A/D board digitizes the picture into sixteen gray levels (zero corresponds to black, and fifteen corresponds to white). Thus, four bits are needed to represent one pixel. The A/D converter outputs two pixels at a time, as an eight bit word. Since the Data General computers use sixteen bit words, the fortran subroutine that activates the Cromemco reads two of the eight bit words and creates a sixteen bit word that is written to disk. What results is a sixteen bit word with four pixels stored in it. (See figure 3.1.)

If any operation is to be performed on the pixels separately, they first have to be "unpacked" from the sixteen bit word. After any processing, the four adjacent pixels



Representation of four pixels in a 16-bit word.
Fig. 3.1

must be "repacked" if the video scene is to be displayed on a standard output device. Also note that the packed version of a video file is an efficient way to store the picture.

Even though the frame is best stored as packed pixels, after processing, the pixels are often no longer four bit integers. For example, after Fourier transformation, the pixels are complex numbers; their magnitudes are real numbers--often greater than 15. Thus, in general, it is easier to store transformed versions of a scene in an unpacked form.

The algorithm presented in this thesis is actually seven stand-alone programs. There are three major reasons for this. First, the 2D-DFT and 2D-IDFT programs were supplied as separate, executable programs. This immediately separated the algorithm into three parts: (1) before the 2D-DFT, (2) between the 2D-DFT and the 2D-IDFT, and (3) after the 2D-IDFT. Second, core size limitations restricts the length of many programs. Finally, many steps of the algorithm have to be executed only once. For example, once the phase of a

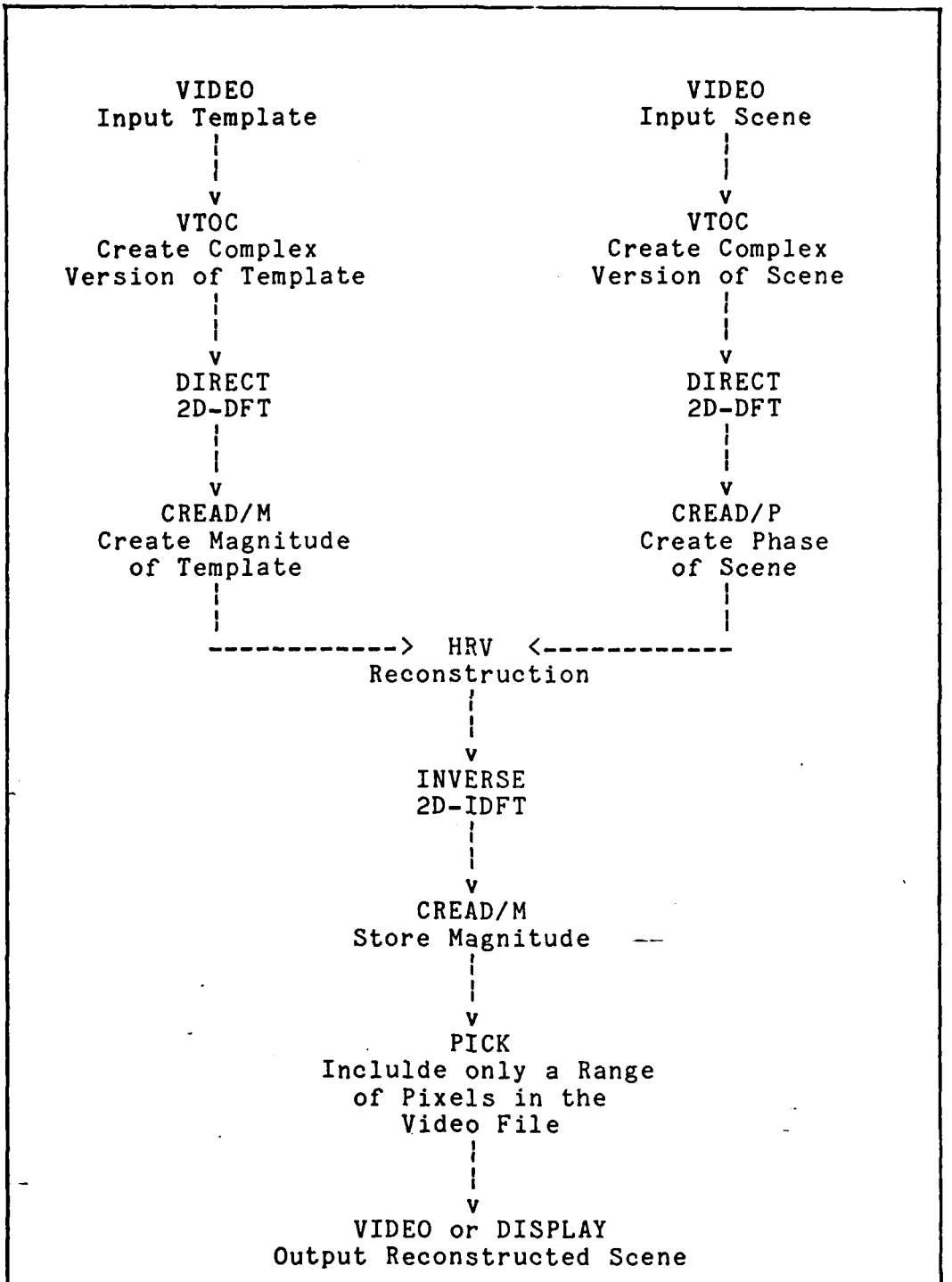
scene is stored (or the magnitude of a template), those data do not need to be created again. Therefore, the algorithm is best executed by separate programs.

Software Development.

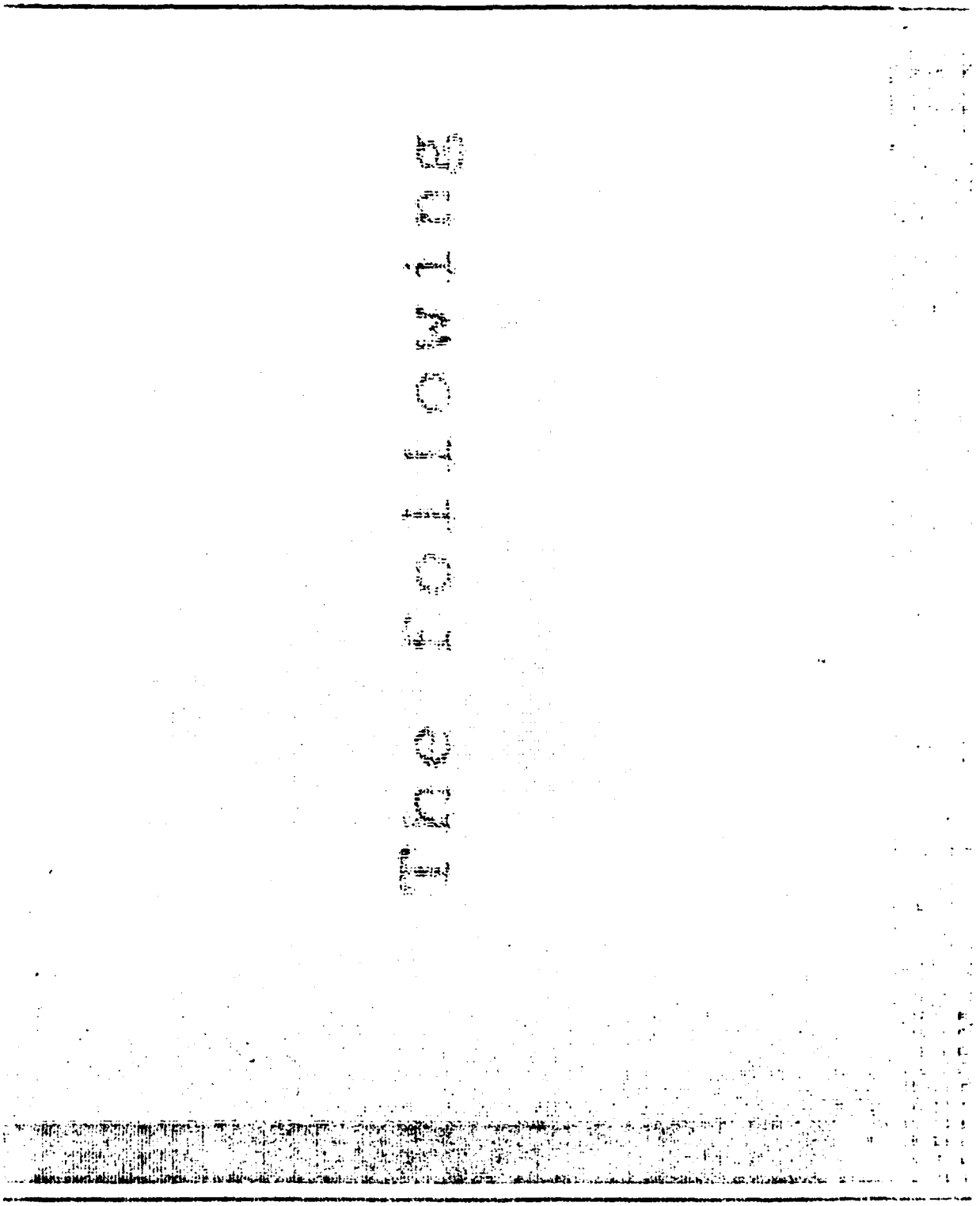
Figure 3.2 contains a flow chart of the programs that make up this segmentation algorithm. The development of each program (except DIRECT and INVERSE, which were supplied programs) is described in general terms in the next paragraphs.

The first of the eight programs is called "VIDEO." This program is a general-purpose video I/O program that sets up the parameters for CHANNEL--the subroutine that activates the Cromemco computer. Certain parameters are always constant for video I/O, but others must be given by the user: input or output, monitor display time, and disk filename. Program VIDEO requests these and then calls subroutine CHANNEL.

The first program that actually performs any processing on the data file is called "VTOC" (Video TO Complex). This program performs three steps: (1) It reads from a video file and unpacks the pixels; (2) It "blanks out" a portion of the picture if the user requests this operation; and (3) It writes the pixels out to disk as complex numbers (imaginary part zero). The blanking step is necessary since the video digitizer/Cromemco combination does not completely digitize a 256 by 256 picture. Twelve pixels on the left end of each line and the last 23 lines are incorrectly digitized. See figure 3.3. Since this noise will add unwanted frequency



Flow Chart of the Reconstruction Algorithm
Figure 3.2



Noise from Video Digitizer/Cromemco Software
Figure 3.3

components to the Fourier transform, the noise was blanked to a constant level. The video-to-complex transformation is needed so that the video file is readable by the 2D-DFT program. This program (DIRECT) reads and writes complex numbers only. Therefore, the only preprocessing necessary is the transformation from video to complex numbers. (Plus blanking, if the user desires.)

After Fourier transformation, the magnitude of the template must be calculated, as must the phase of the scene. A program to perform these operations is "CREAD" (for Complex READ). This program computes the magnitude (see equation 2.3) or phase (see equation 2.4) of a complex file and writes the output as real numbers to a new disk file. At this point, the complex files need not be saved--no more operations are performed on them, and the original video file is the most easily understood representation of the scene. Their disposal results in a large savings of disk space.

The next program performs a simplified version of Horev's algorithm; the program is named "HRV." It performs the reconstruction of a magnitude file with a phase file, creating a complex file that the 2D-IDFT (INVERSE) can read. Equations 2.5 and 2.6 are used to create the complex terms. The steps performed by program HRV are: (1) Computation of the square root of the sum of the squares of each magnitude component (for energy normalization), (2) Reconstruct the new complex scene from the energy-normalized magnitudes, and (3) Writing the new complex components to disk. After the

reconstruction operation and the 2D-IDFT, the data file is complex. The magnitude of this complex scene contains the visual information, so the scene can be stored much more efficiently in magnitude form. Program CREAD calculates the magnitude of this file; no other computation is performed on the complex numbers.

However, since segmentation was verified in this work by viewing the reconstructed scene, at least one more program had to be written. This program converts the (reconstructed) magnitude file to repacked video pixels for display on a monitor. Two versions of this type of program were written. The first, named PIX1, clips the magnitudes into a zero to fifteen range. This is necessary since the magnitudes of the reconstructed file are typically of the order 0.0002 to 2.0; they had to be linearly scaled to the range zero to fifteen. Equation 3.1 was used to clip the magnitudes to the proper range.

$$PIXEL_i = 15 * [(Rmax - MAG_i)/(Rmax - Rmin)] \qquad 3.1$$

In equation 3.1, MAG_i is the magnitude of the pixel after the CREAD, Rmax is the largest magnitude, and Rmin is the smallest magnitude. Clipping occurs when the magnitude is outside of a user-specified range. If the magnitude is above the range, the pixel is set to 15. If the magnitude is below the range, the pixel is set to zero. Clipping the magnitudes does not properly perform segmentation, however, since the magnitudes that are clipped to zero will be

indistinguishable from the black (zero) pixels that make up the segmented character. Thus, PIX1 had to be changed from "clipping" to "exclusion"; pixels outside the range were not printed by the new program. Only the pixels that fall into the specified range are printed. If the proper range has been selected, then only the segmented character will be visible. This program was named "PICK", and the results presented in the next chapter were created with program PICK, except that 11 was used instead of 15 in equation 3.1 so that there would be a pixel level "jump" from the white background (excluded pixels) to the character pixels.

Another useful program that is also not in the algorithm is called DISPLAY. This program was written to print a video scene on a Printronix P-300 lineprinter. This program converts the pixels to arrays of lineprinter dots that represent the sixteen gray levels. The video figures included in this thesis were printed from video files via DISPLAY.

With this software developed, different sets of characters were processed. The results are given in the next chapter.

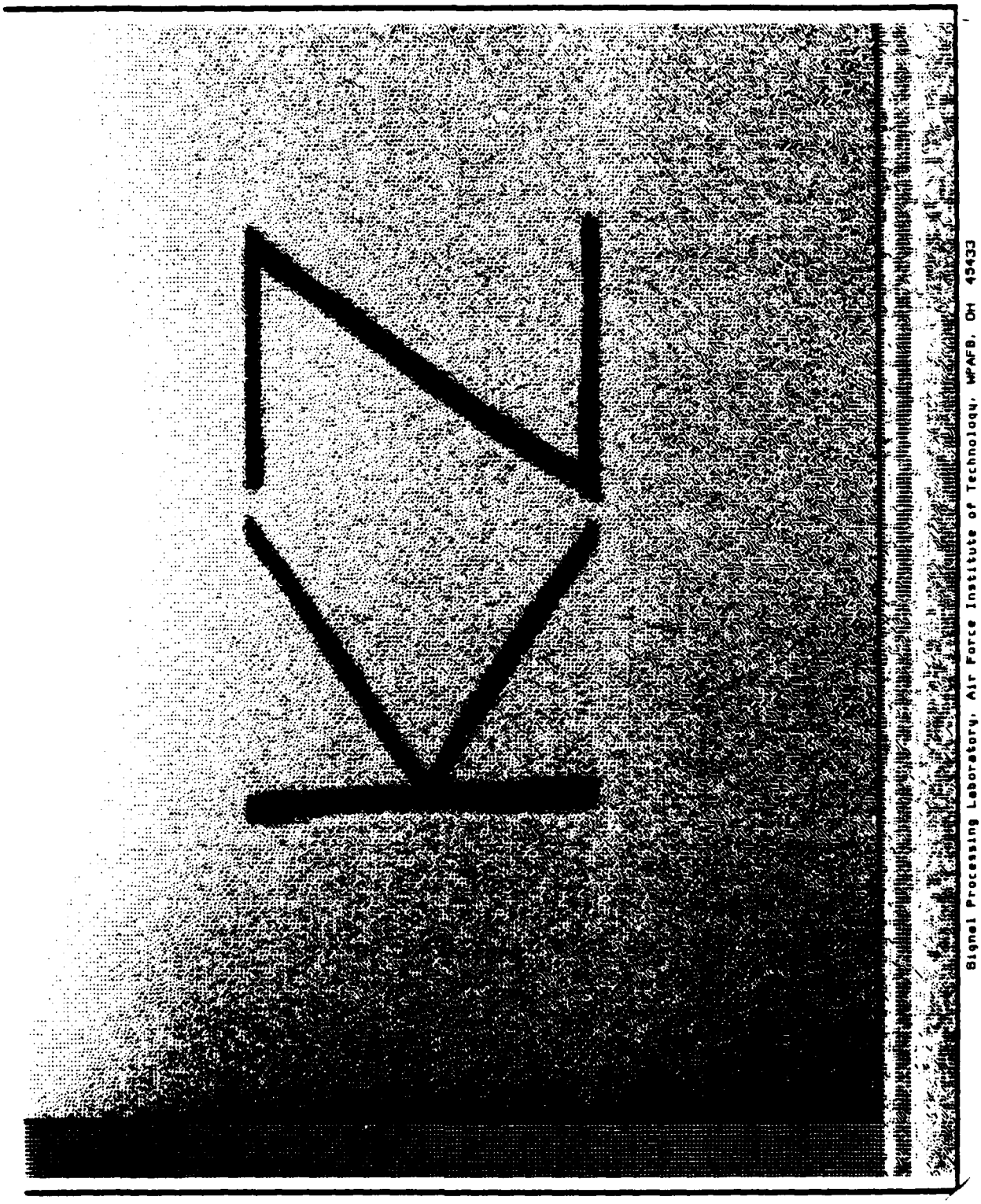
IV. Results of the Reconstruction Algorithm

General.

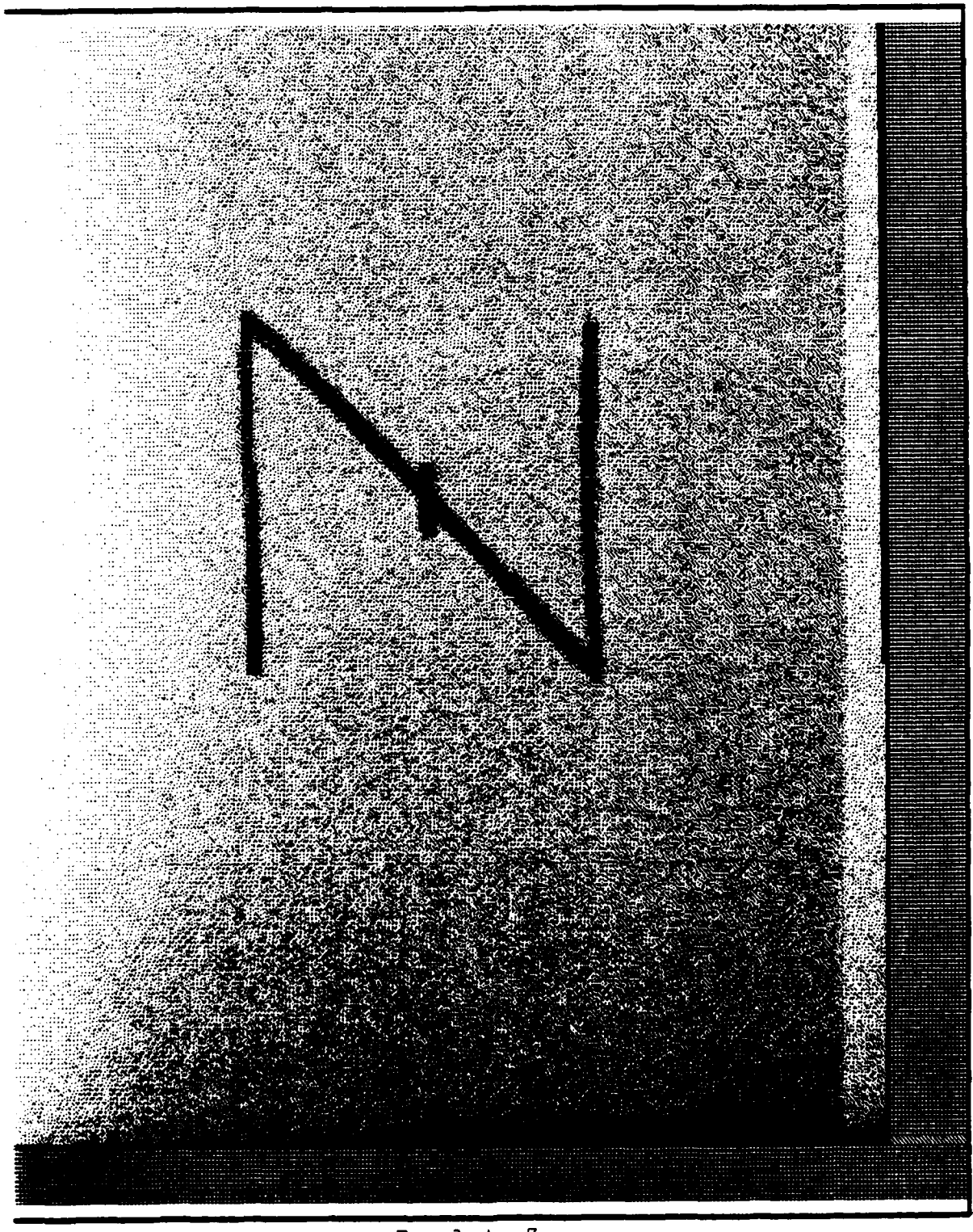
The last two chapters have developed the theory and given a brief description of the software for the reconstruction algorithm. This chapter contains some of the results achieved when segmenting English characters. The complete alphabet is not used, since there was not enough time to process all of the possible two or three letter pairs. Most of the characters presented were handwritten by one of the students in the SPL. The only requirements placed on the size or shape of the characters is that they be of a specified height and that they be nonscript. The specified height served two purposes: first, it forced the characters to fill up a large part of the window (as described on pages 8 and 10 in chapter 2), and second, it was assumed that an optical zoom feature can be implemented to insure that all characters are of approximately the same height. The nonscript requirement was to keep the style of each character to the general, standard shape.

Segmentation Results.

The first segmentation results presented are of the scene "KZ" (figure 4.1). The phase of this scene was calculated, and then the magnitude of "Z" (figure 4.2) was calculated. These were recombined, inverse Fourier transformed, and



Scene KZ
Figure 4.1

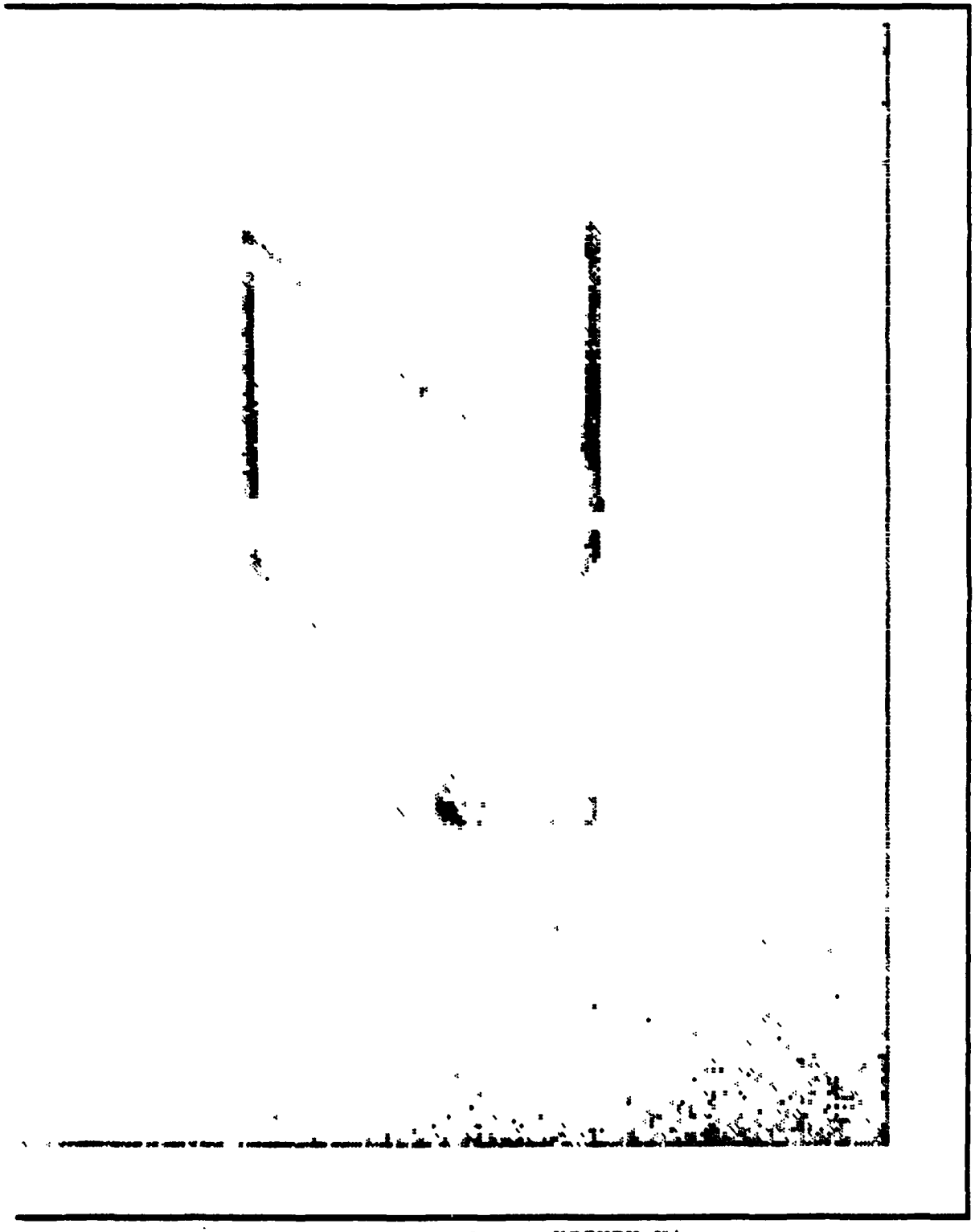


Template Z
Figure 4.2

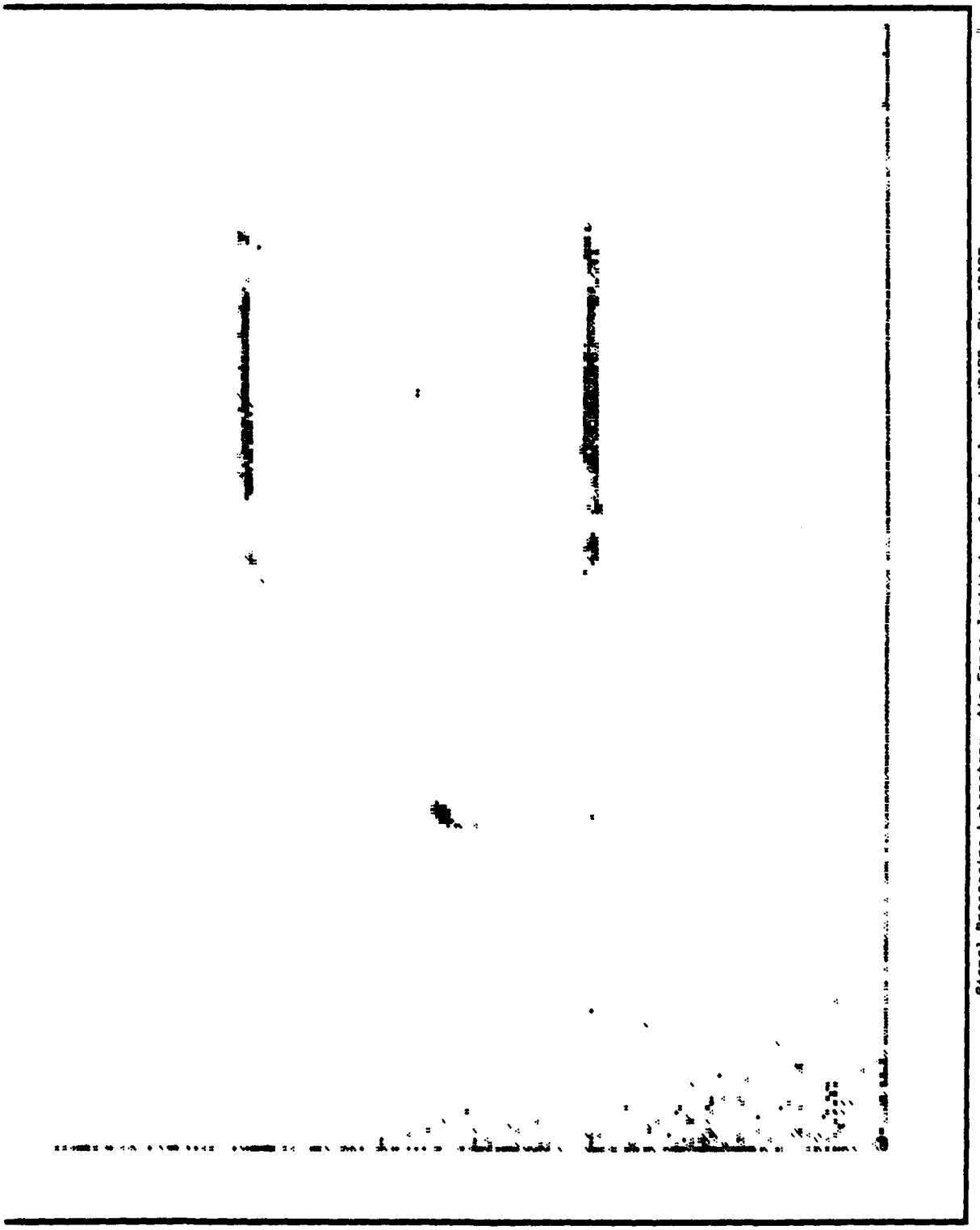
then the magnitude of the recombined scene was saved. Program PICK was used to print three lower ranges of pixel levels. KZZHRV.V4 (figure 4.3) has a maximum of 0.2910 and a minimum of 0.0. Recall that the magnitude range after the inverse Fourier transformation is 0.0002 to 2.0. (A note on file notation: the ".V4" means that the file is video file number four; the "HRV" means that the recombination process has been completed; the character in front of "HRV" is the template; and the characters in front of the template character are the scene characters.) In figure 4.3, a portion of the "K" is present, while the "Z" still has its whole width present. Dropping to a lower pixel range (figure 4.4) removes more of the "K", while the "Z" width is still completely present. (The maximum magnitude printed is 0.2190.) Dropping to a still lower maximum magnitude, KZZHRV.V2, we see that the "K" is virtually removed, while the left-to-right range of the "Z" is easily seen (figure 4.5). The maximum level is 0.1460, with the minimum still 0.0.

Next, "K" was used as a template to remove the "Z" (see figure 4.6). KZKHRV.V3 (figure 4.7) has a maximum magnitude of 0.2120 and a minimum of 0.0. The "Z" is virtually gone. Dropping to a maximum of 0.1410 (figure 4.8) or 0.07060 (figure 4.9) shows that the "K" is still present, while there is no width to the "Z" at all. Even if the remaining "Z" pixels of KZKHRV.V1 (figure 4.9) are used in bounding "K", there will be little error in the placement of the window around "K."

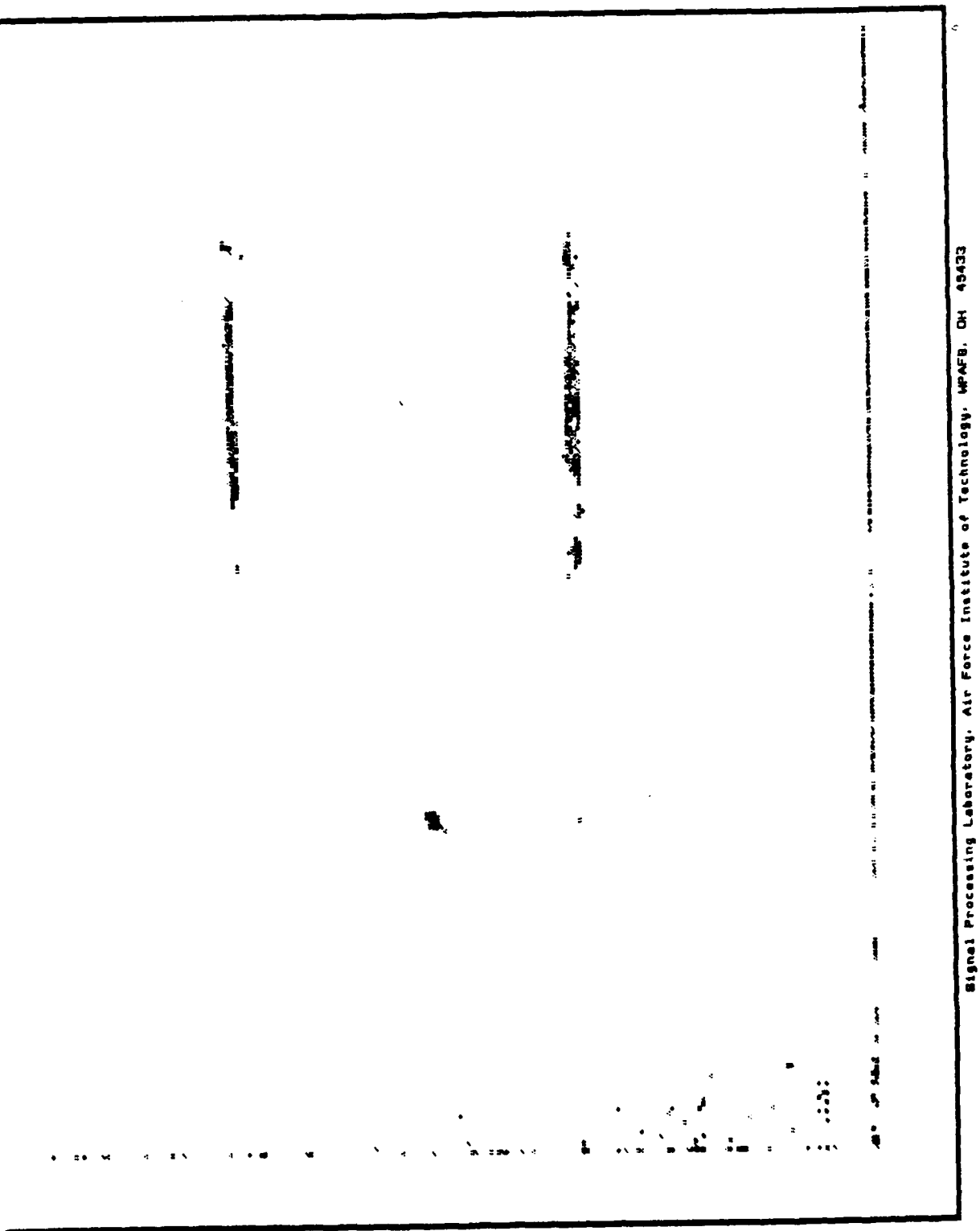
The characters "K" and "Z" both have a lot of diagonal



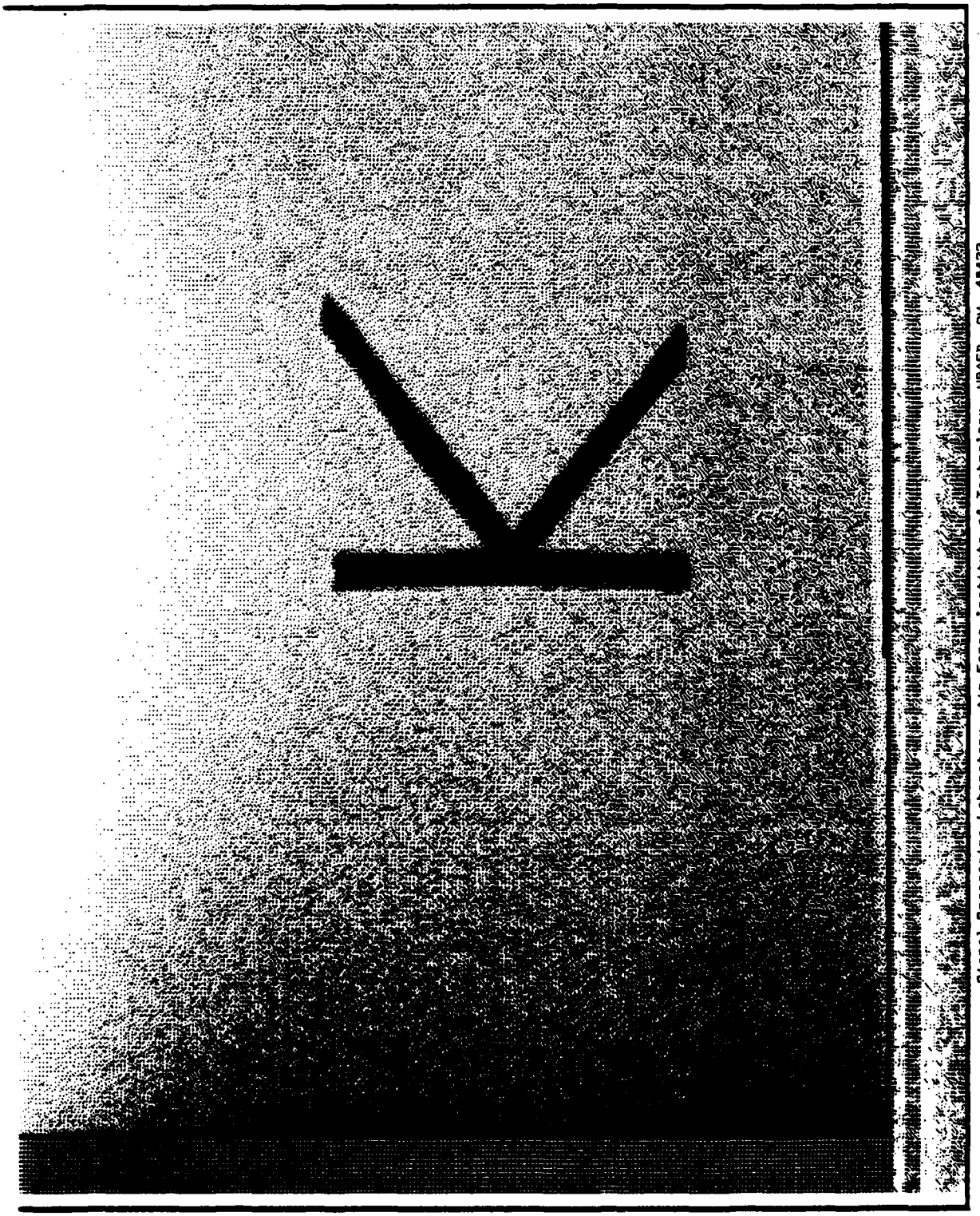
Reconstructed Scene KZZHRV.V4
Figure 4.3



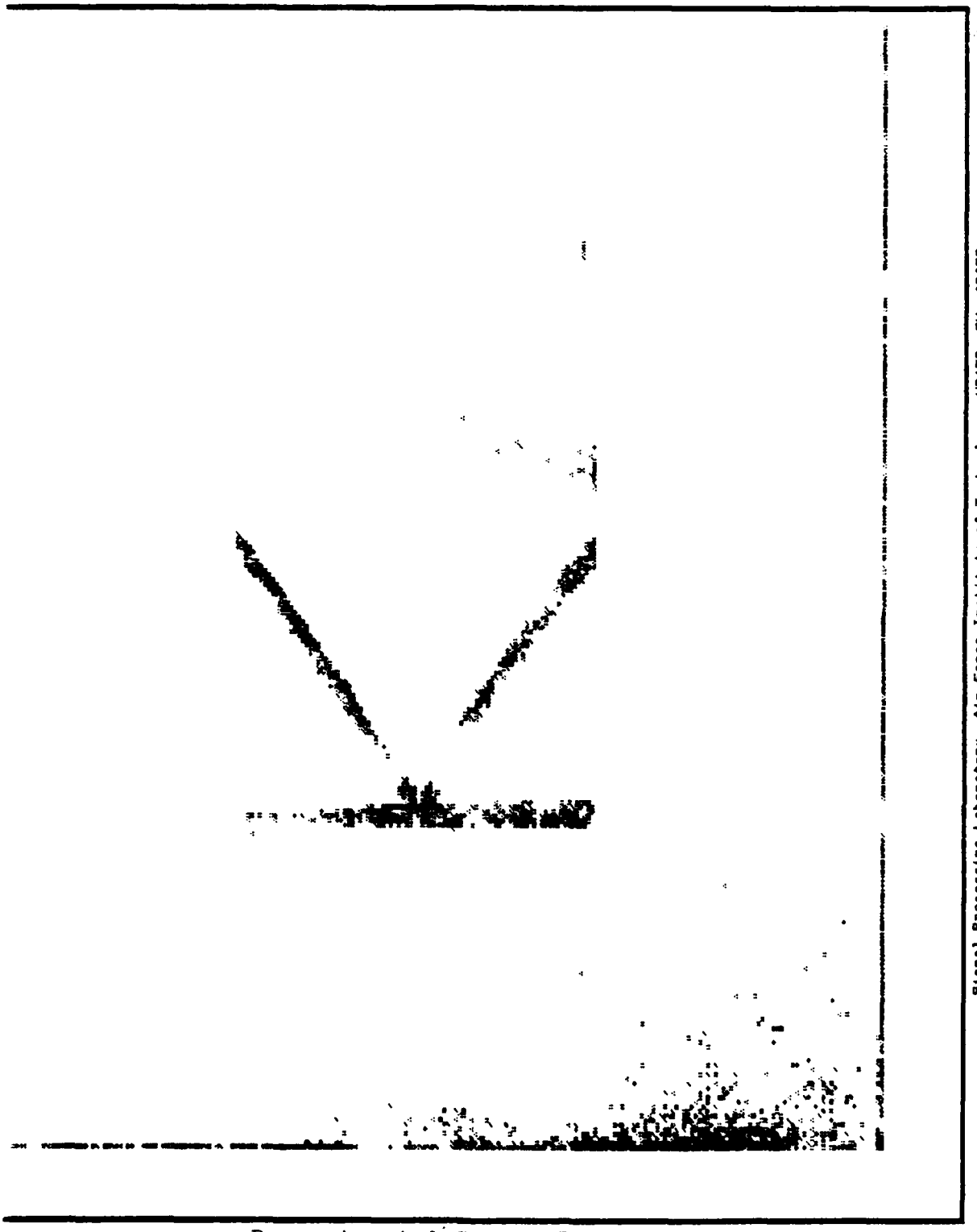
Reconstructed Scene KZZHRV.V3
Figure 4.4



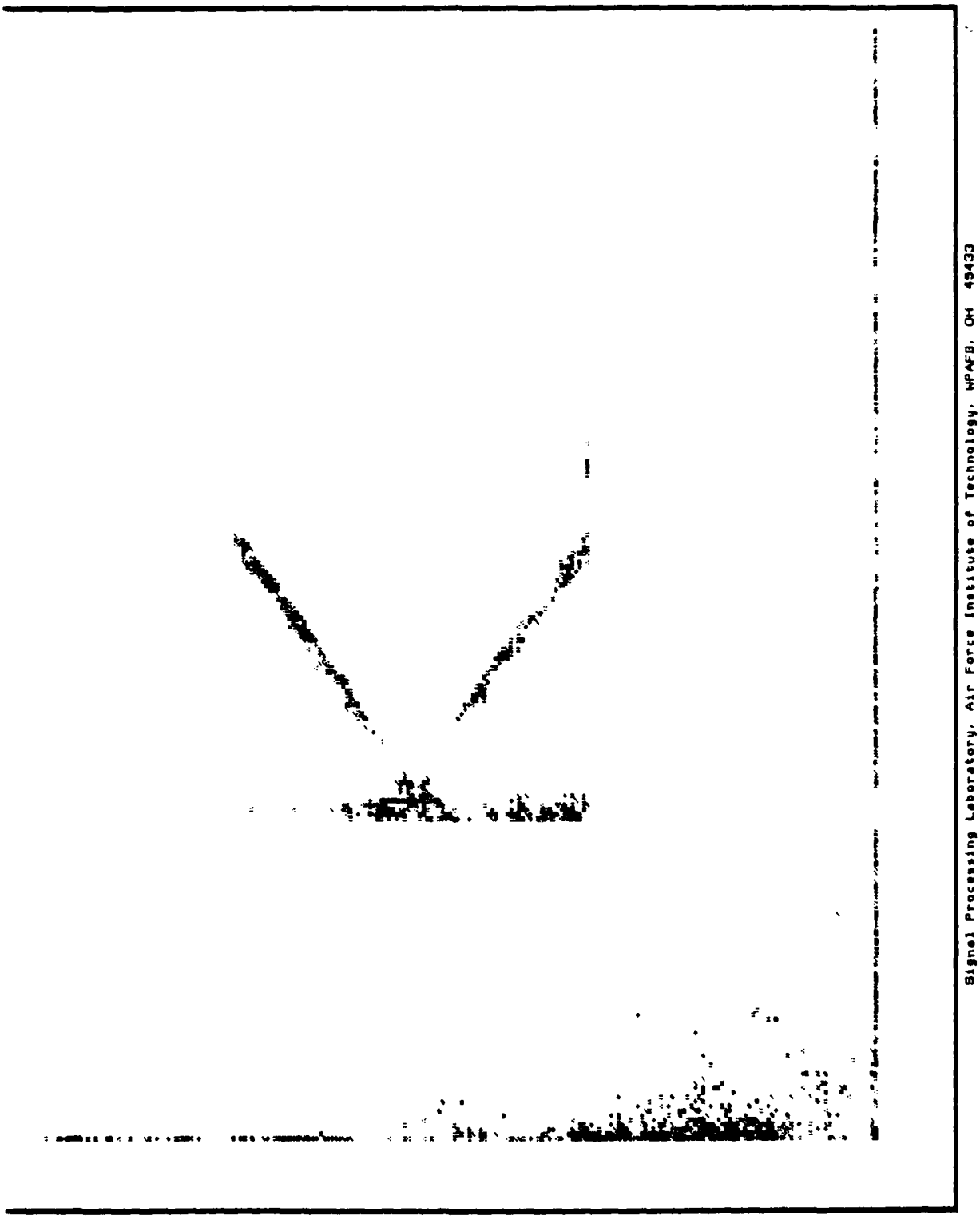
Reconstructed Scene KZZHRV.V2
Figure 4.5



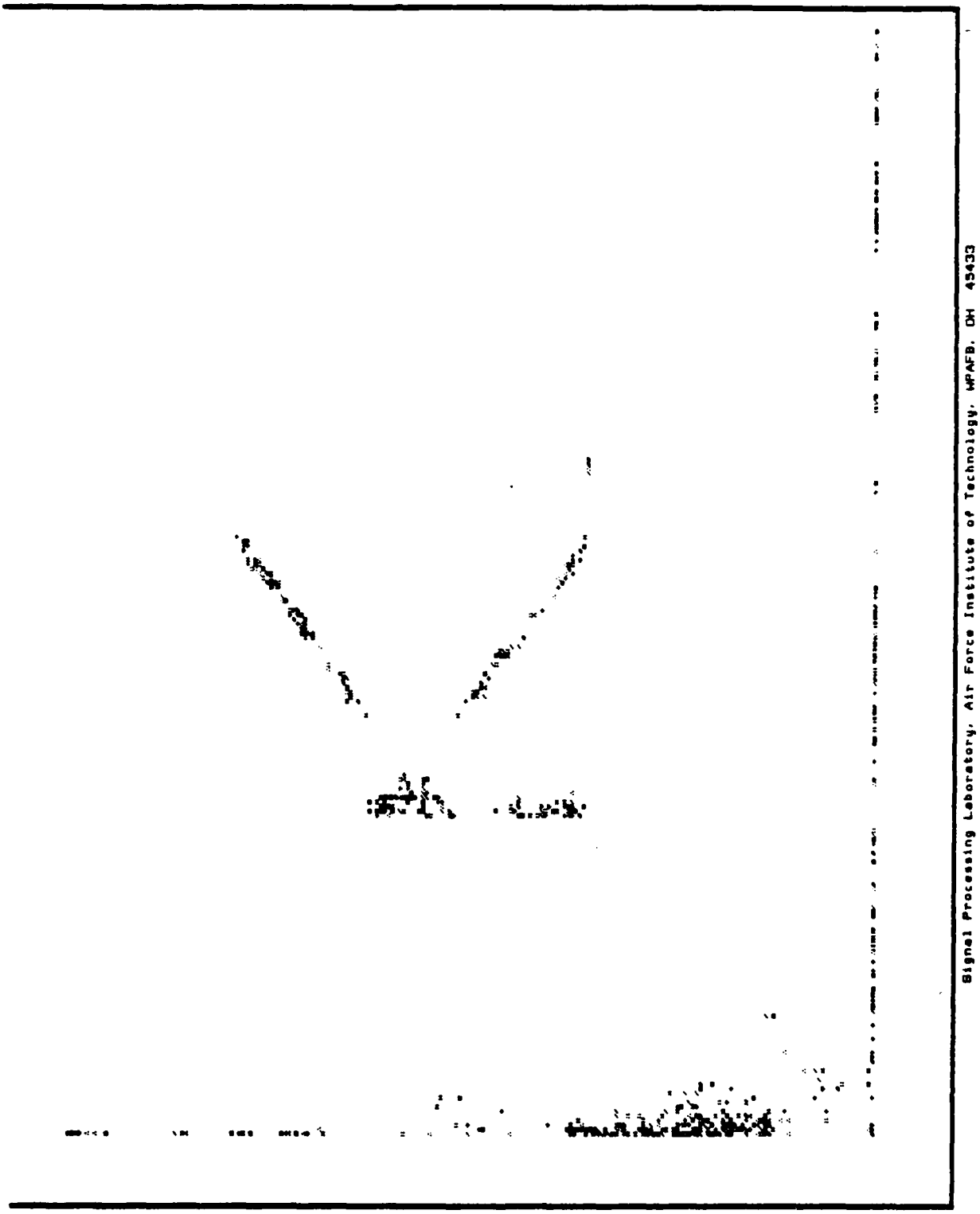
Template K
Figure 4.6



Reconstructed Scene KZKHRV.V3
Figure 4.7



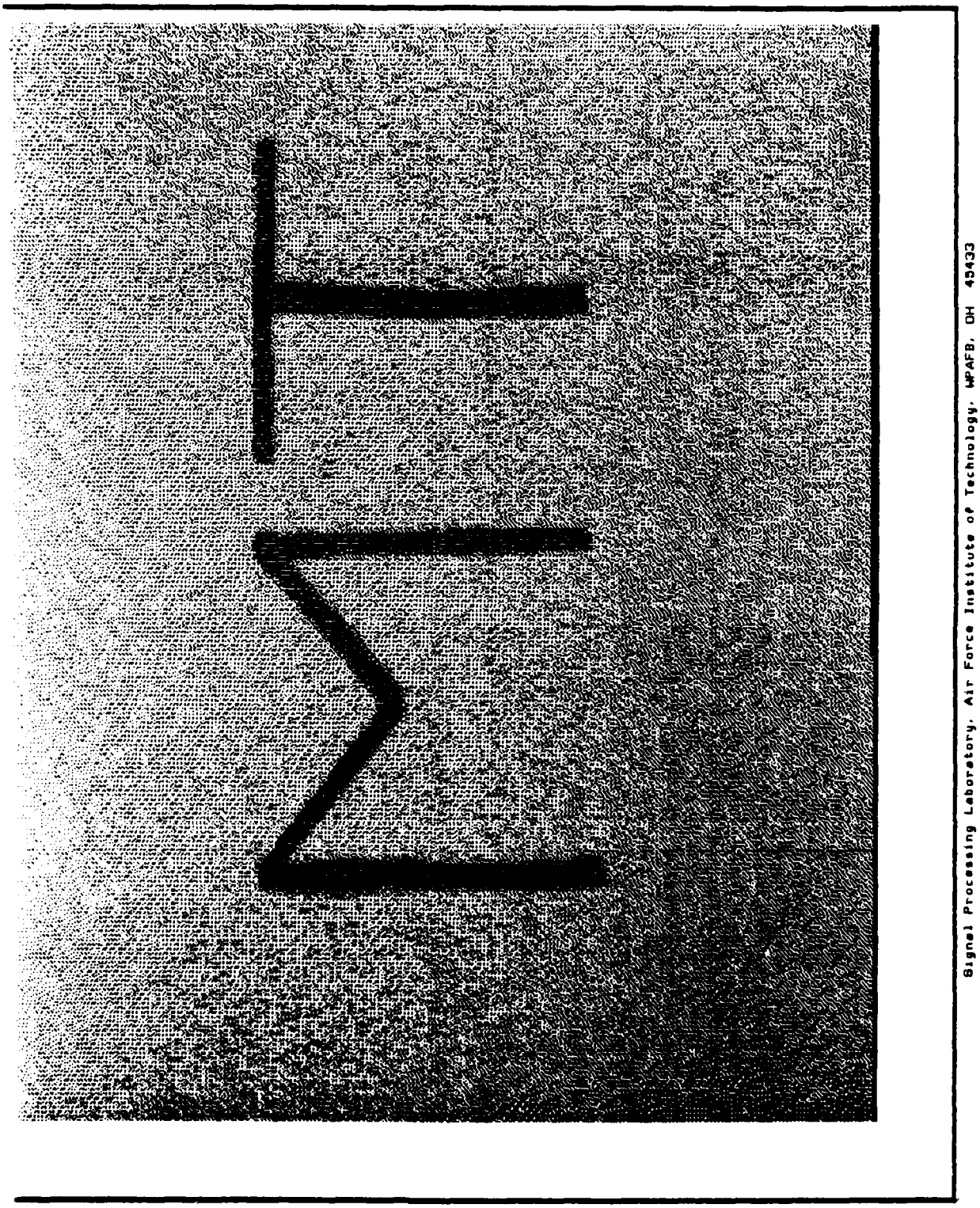
Reconstructed Scene KZKHRV.V2
Figure 4.8



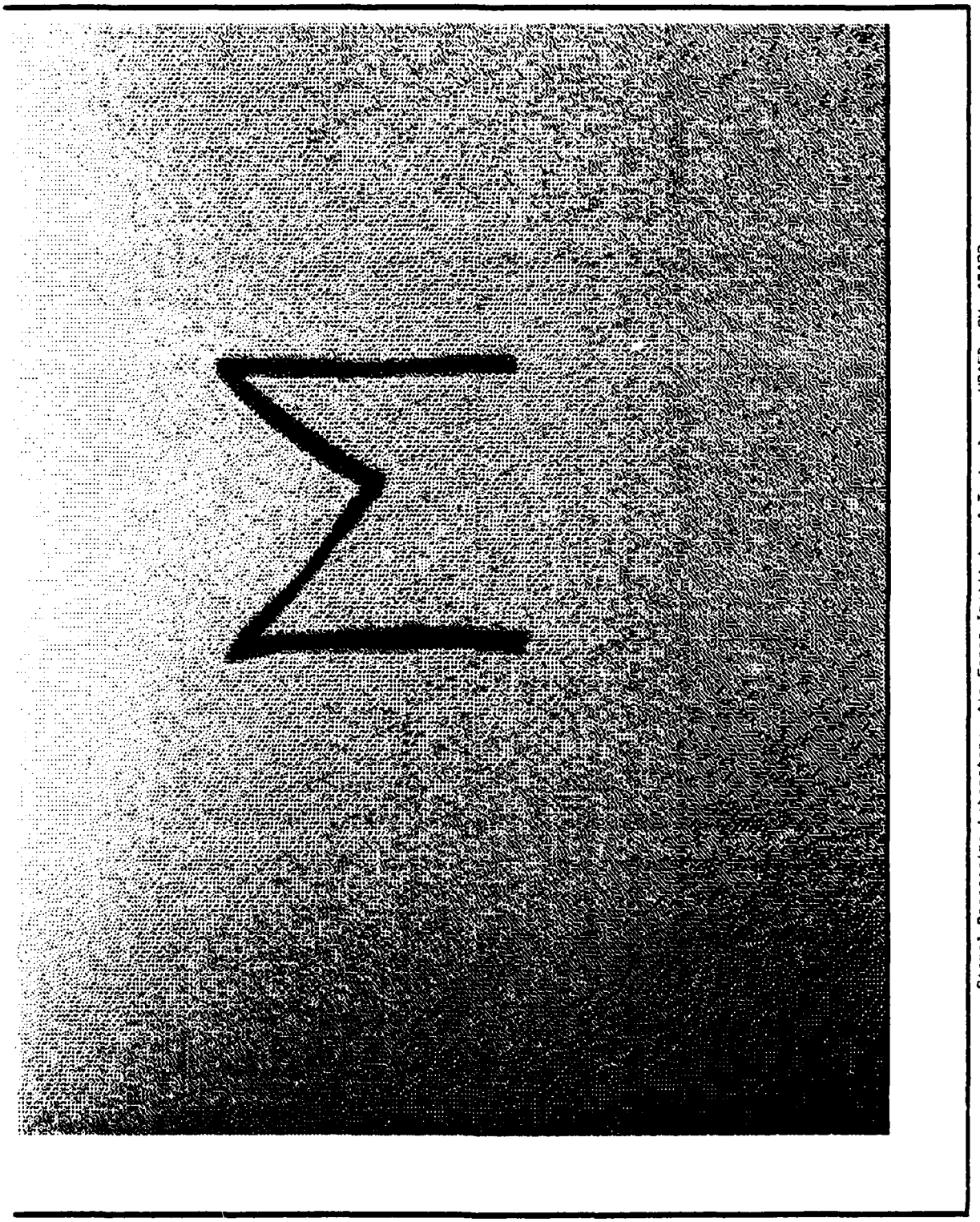
Reconstructed Scene KZKHRV.V1
Figure 4.9

information in them, but the reconstruction process displays only the diagonal information that is contained in the template; the diagonal in "Z" is at a different angle than that in "K." The "Z" is segmented mostly by its horizontal information, while the "K" is segmented by its vertical information.

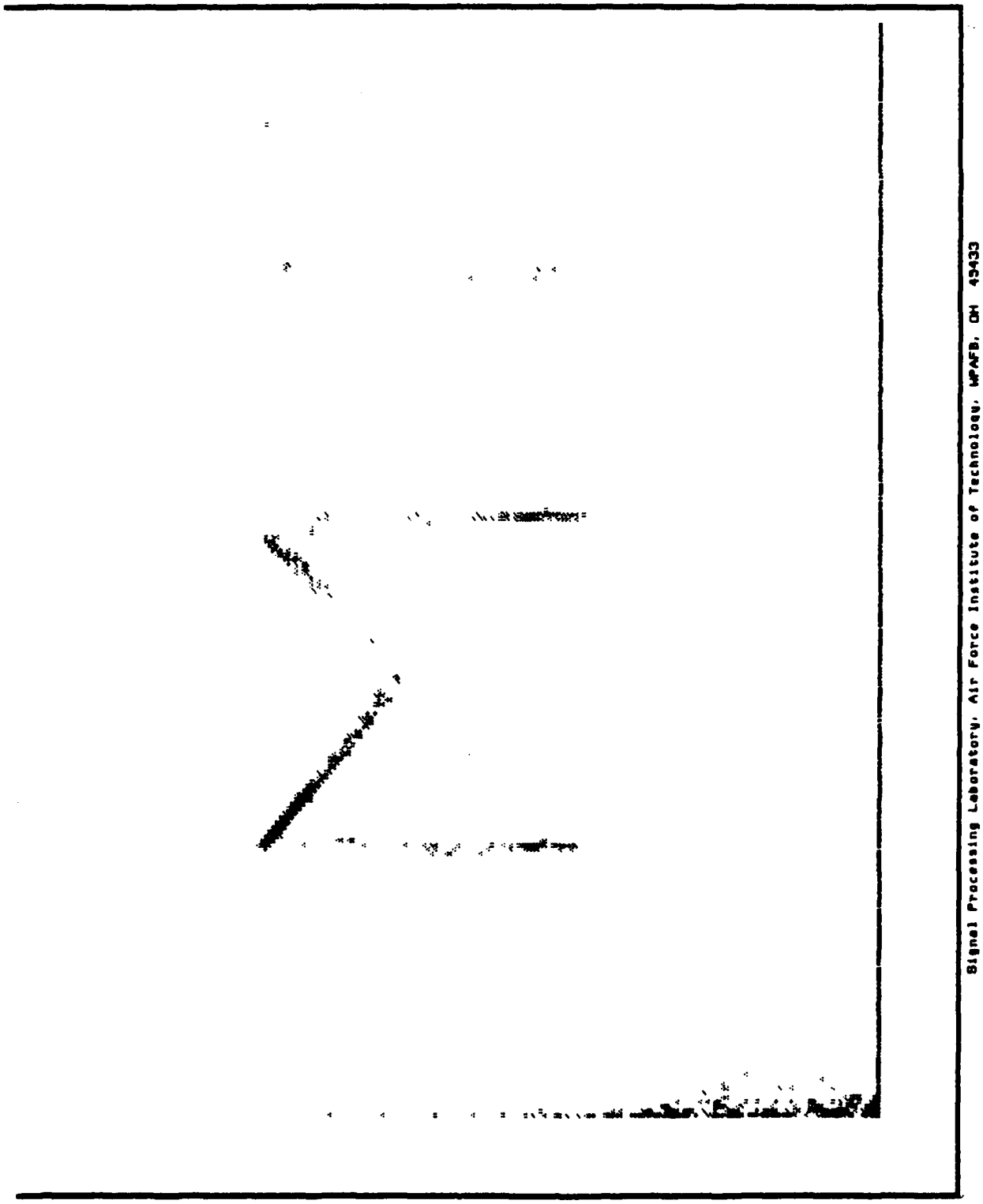
The next set of reconstruction results is with the letter pair "MT" (figure 4.10). The magnitude of the template character "M" (figure 4.11) was calculated and used to segment the "M" from the "T"; MTMHRV.V5 (figure 4.12) has a maximum pixel level of 0.3370 and a minimum pixel magnitude of 0.0. The few sparse pixels of the "T" could give a threshold algorithm some difficulty, but dropping the maximum magnitude level via PICK down to 0.2700 gives us MTMHRV.V4 (figure 4.13). The "T" is completely removed. To be completely general, "T" was then used as a template (figure 4.14) to remove the "M." Both "M" and "T" have a lot of vertical information, and MTTHRV.V5 (figure 4.15) shows that all of the vertical strokes in the characters are present. The maximum magnitude included is 0.3440, with the minimum still 0.0. Dropping the maximum level to 0.2750 creates MTTHRV.V4 (figure 4.16). The crossbar on the "T" is still completely present, but the right-hand vertical stroke in the "M" would probably be sensed by a threshold algorithm. Dropping the maximum to 0.2070 in MTTHRV.V3 (figure 4.17) begins to remove some of the crossbar of the "T", while the right-hand bar in the "M" is still present. However, in either figure 4.16 or 4.17, we can see that even if the right-hand vertical pixels of the



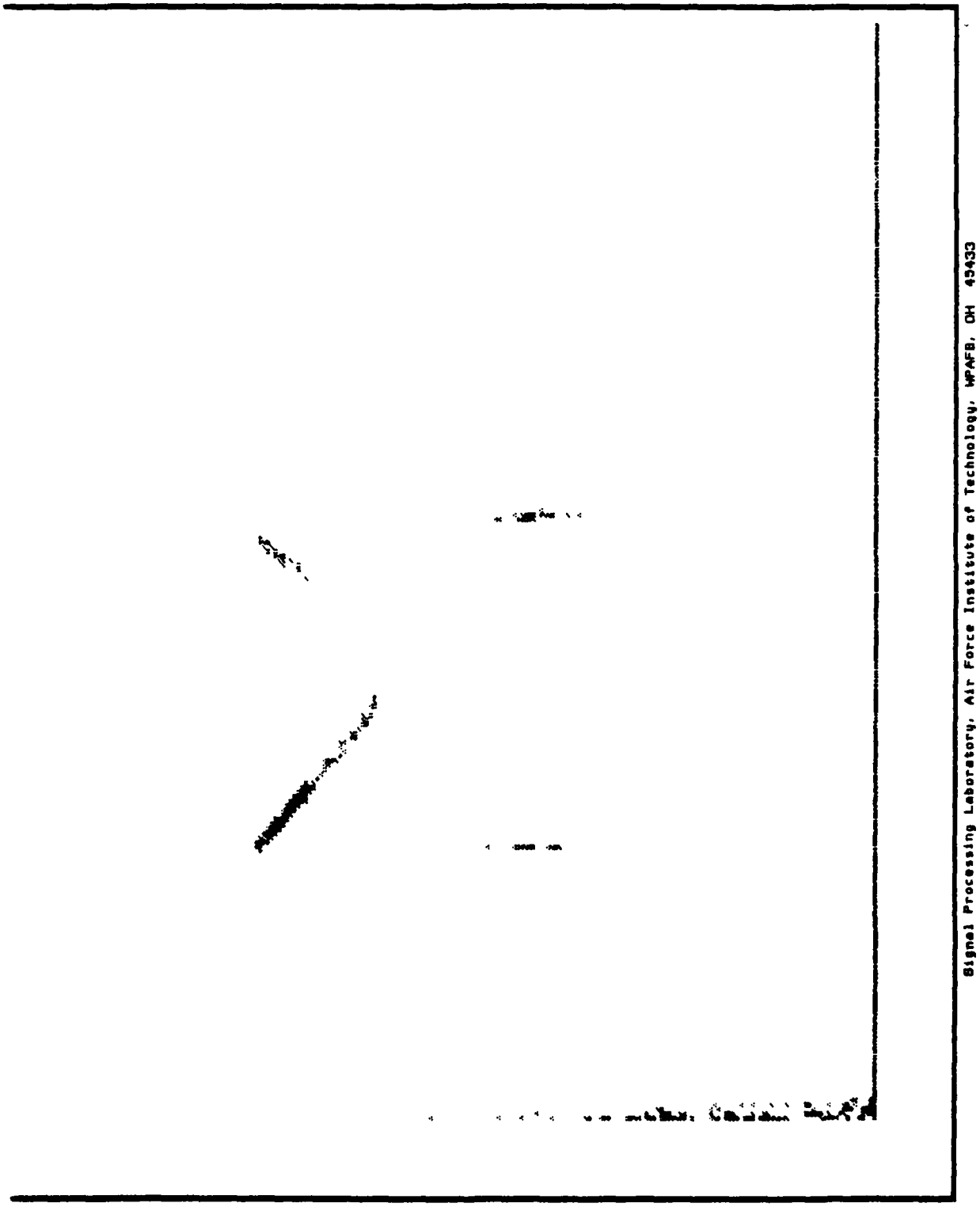
Scene MT
Figure 4.10



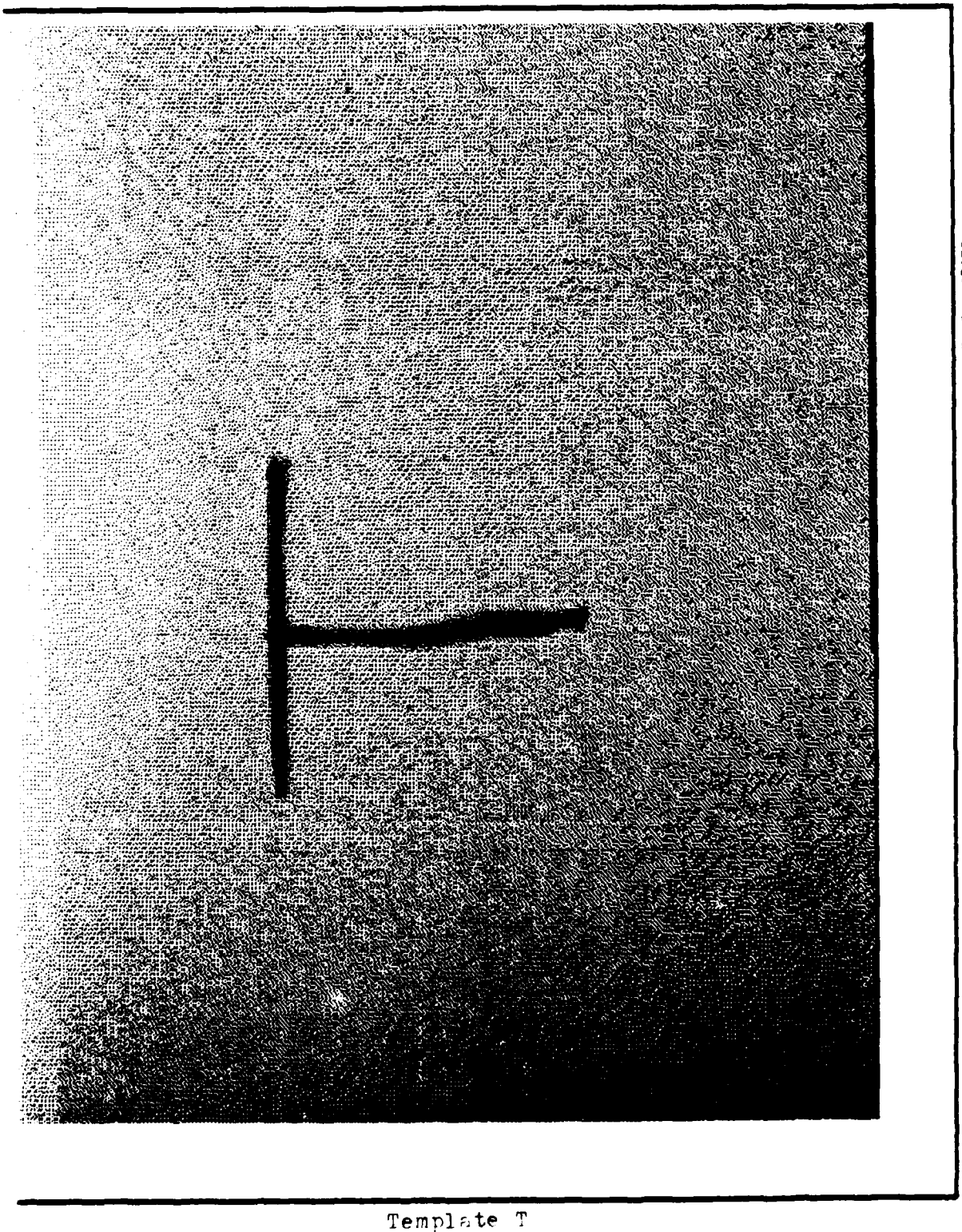
Template M
Figure 4.11



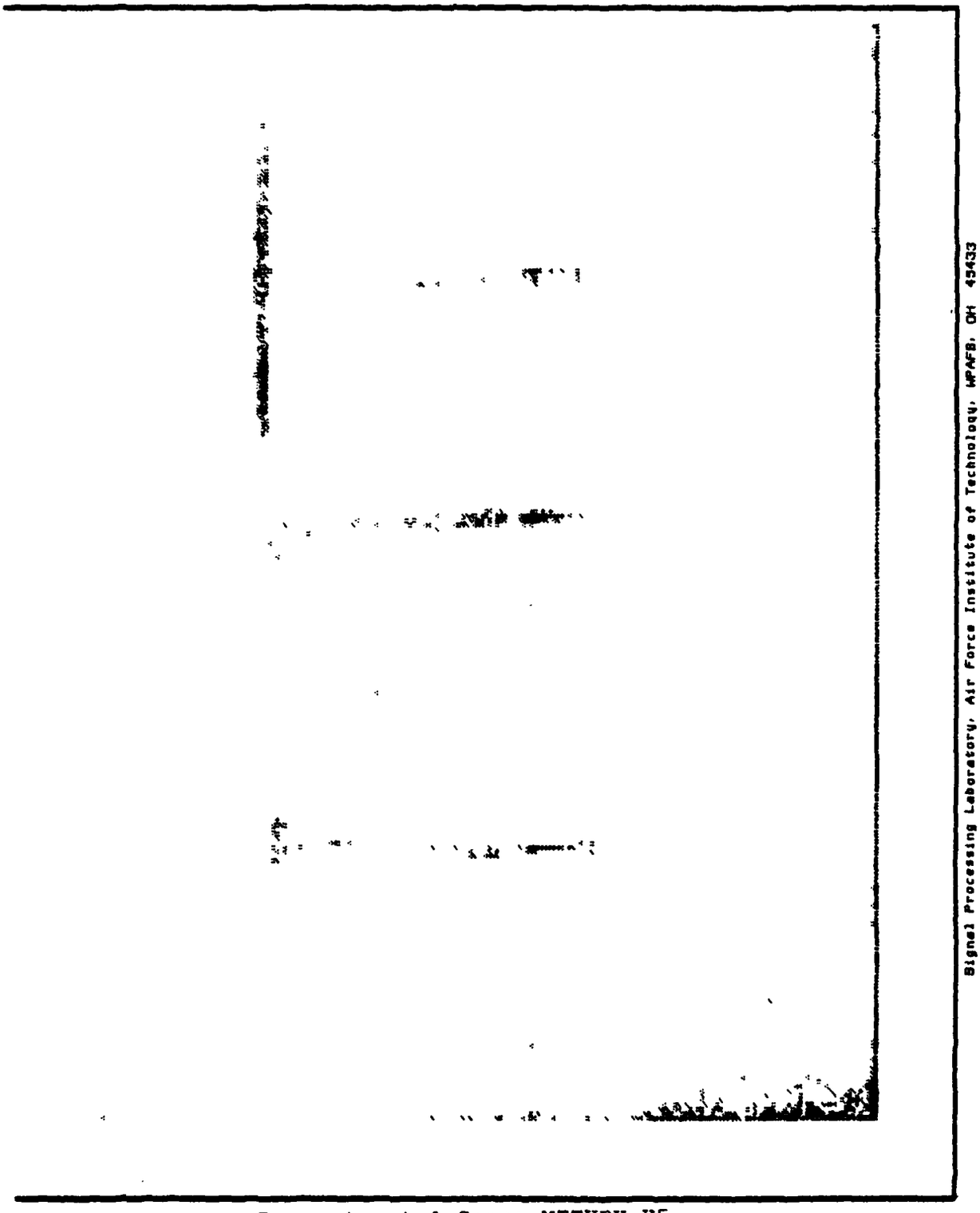
Reconstructed Scene MTMHRV.V5
Figure 4.12



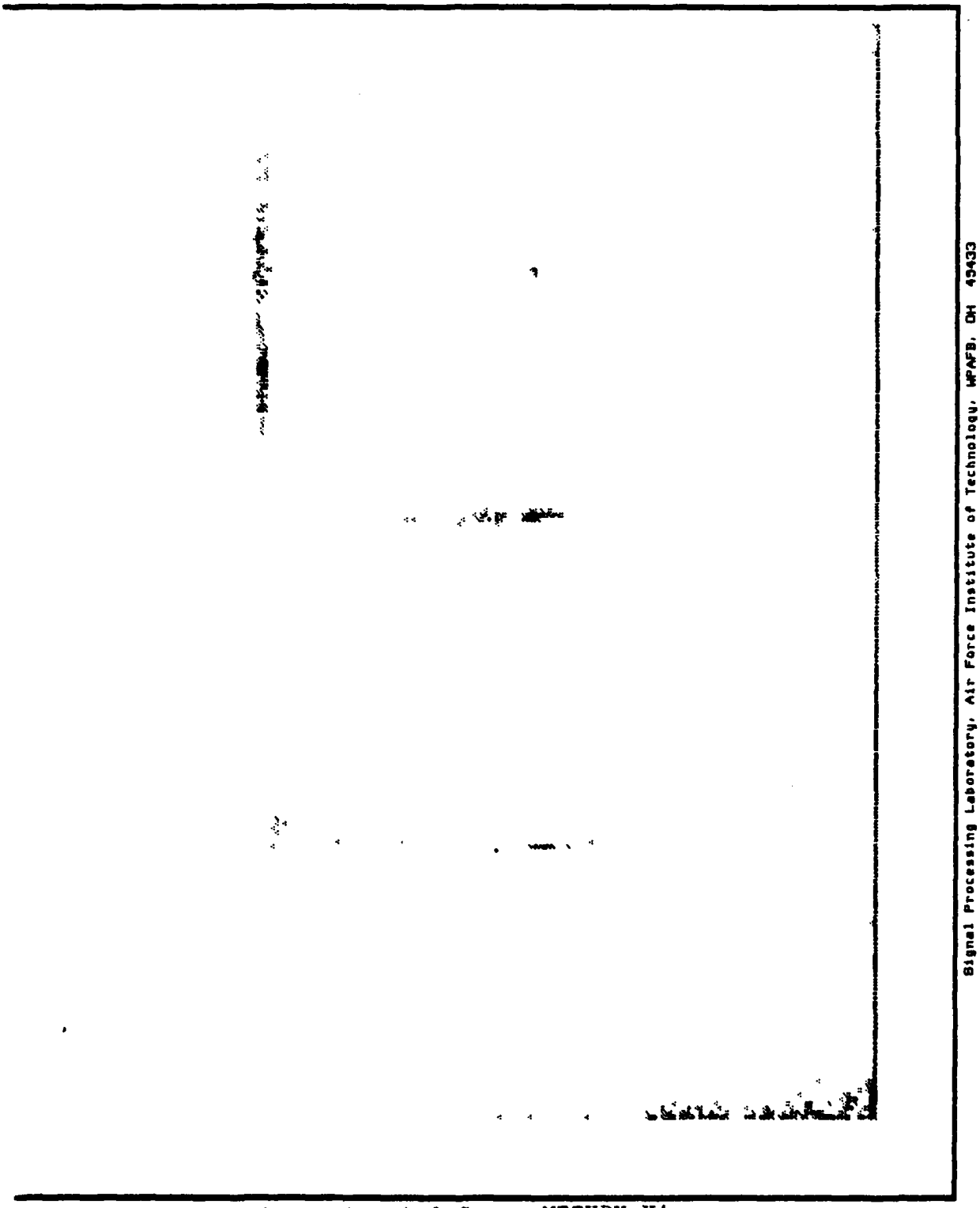
Reconstructed Scene MTMHRV.V4
Figure 4.13



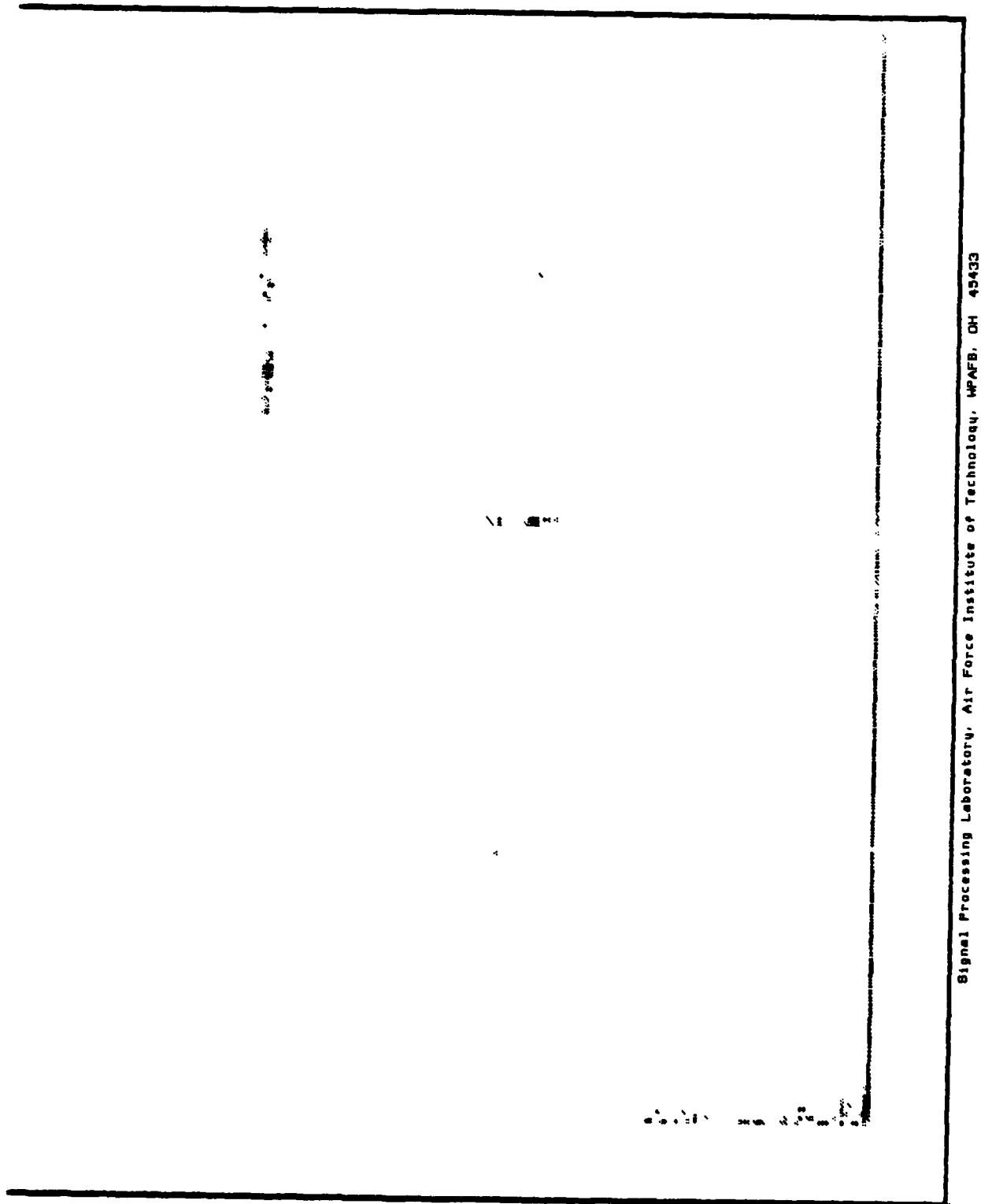
Template T
Figure 4.14



Reconstructed Scene MTTHRV.V5
Figure 4.15



Reconstructed Scene MTTHRV.V4
Figure 4.16



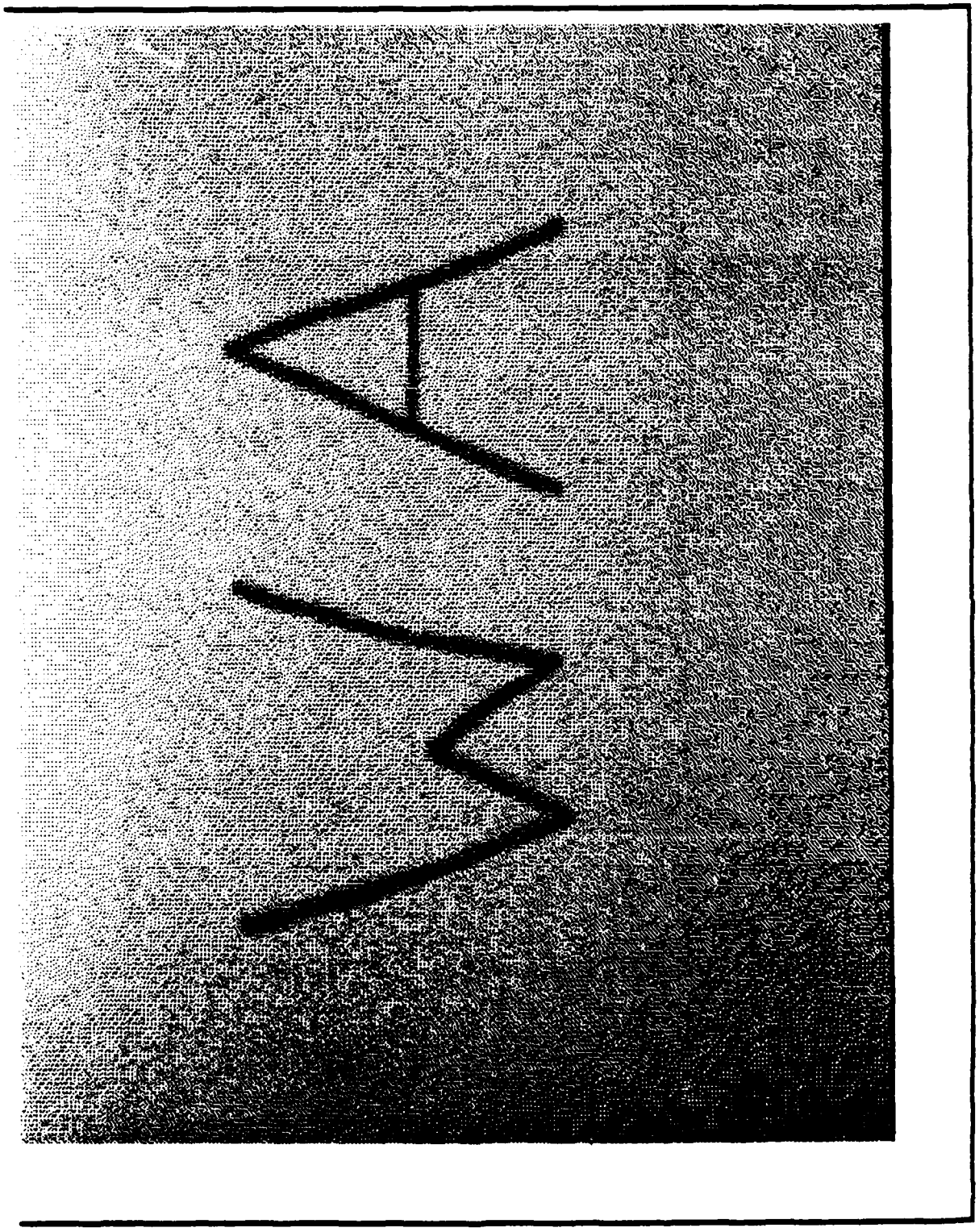
Reconstructed Scene MTTHRV.V3
Figure 4.17

"M" were included as part of the "T" (by thresholding), the character "T" will still be adequately segmented.

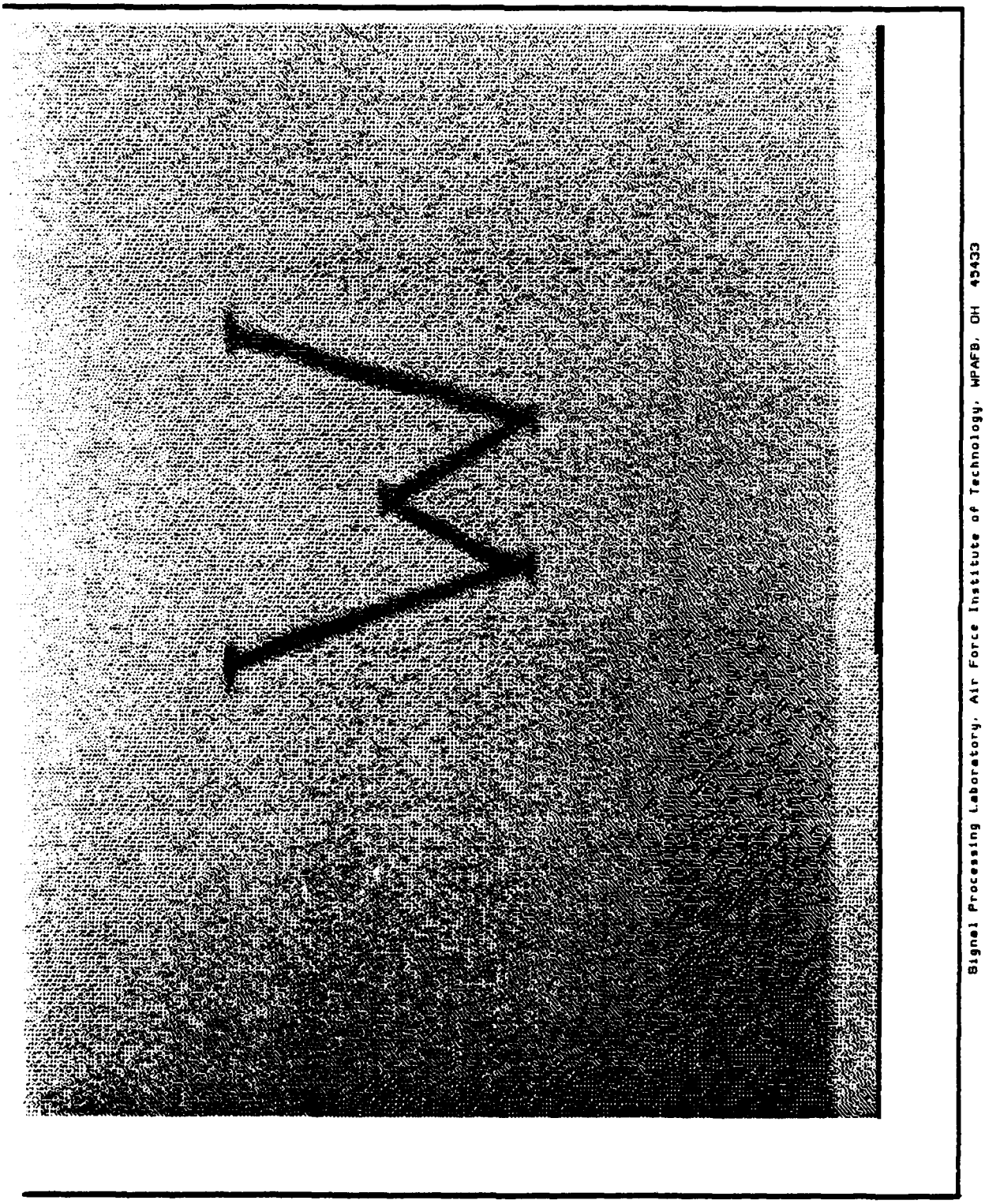
A pair of characters that is much more difficult to segment is shown in figure 4.18. Both the "W" and the "A" have many similar features. To further stress the algorithm, serifs were included on the template "W" (figure 4.19). The peak of the "A" and the center of the "W" both are the same: a "^". WAWHRV.V2 (figure 4.20) has a maximum level of 0.3670 (minimum 0.0), and the "W" is very much present; the "A" is not removed enough to segment the "W", however. In the next figure (4.21), WAWHRV.V3 has its maximum down to 0.3058, and the "A" is now just two points. The pixel levels remaining in the "W" are higher than those in the "A", but both characters may still meet a threshold.

Using an "A" as a template (figure 4.22) should give similar results, and it does. In figure 4.23 WAAHRV.V2 (maximum is 0.1480, minimum 0.0) shows that the "/" is located in the "A" and the "\" is found in the "W". Dropping to a lower maximum magnitude removes much of the "W" (figure 4.24), but both characters would most likely meet a threshold criterion. (Dropping to a level that does not include the "W" also does not include the "A", since they have the same maximum magnitudes before conversion to pixels.) The maximum magnitude was dropped to 0.0738. If both characters were included in the segmentation window, the correlation process would pick out the "A."

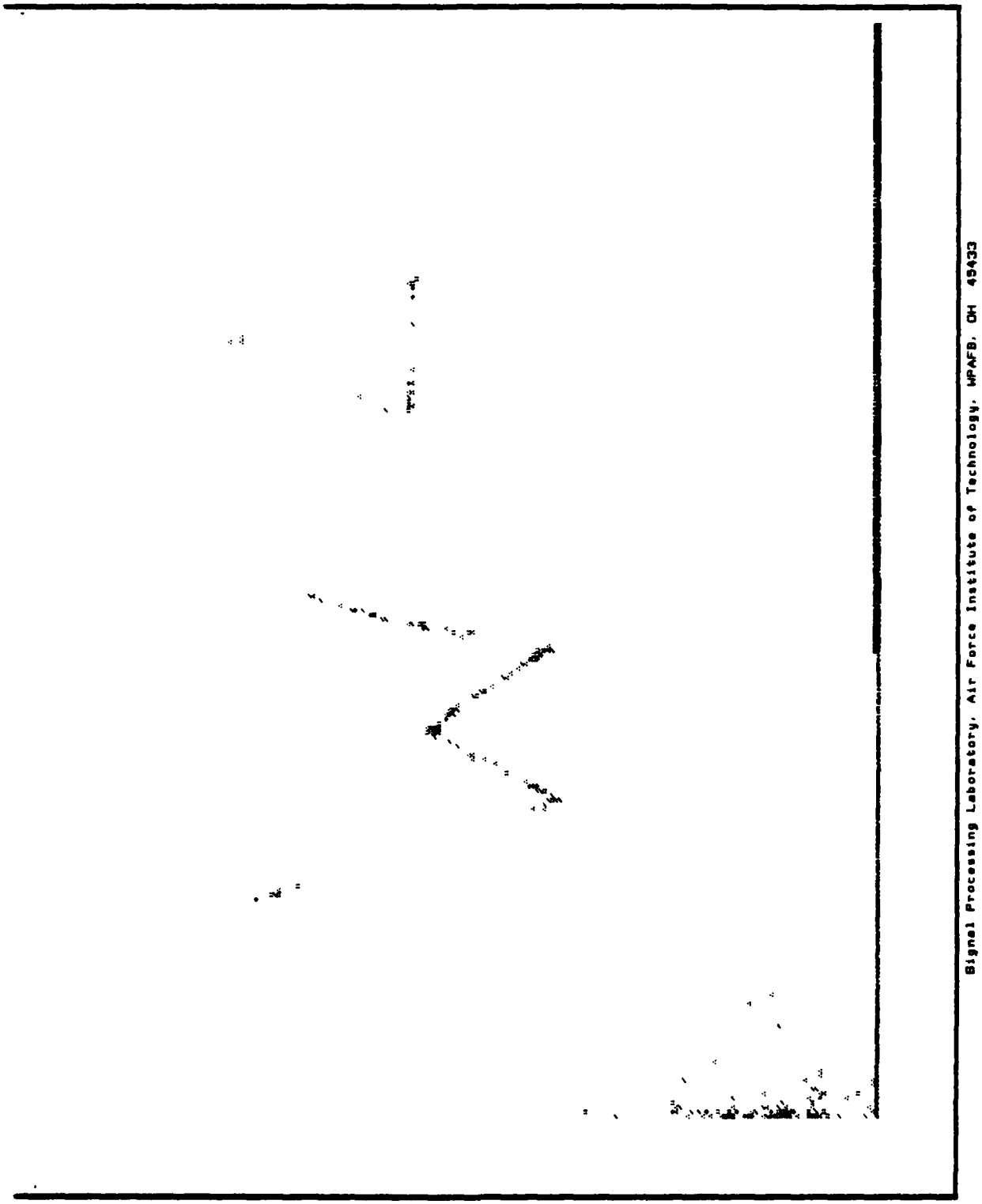
An even more similar pair of characters is "O" and "Q."



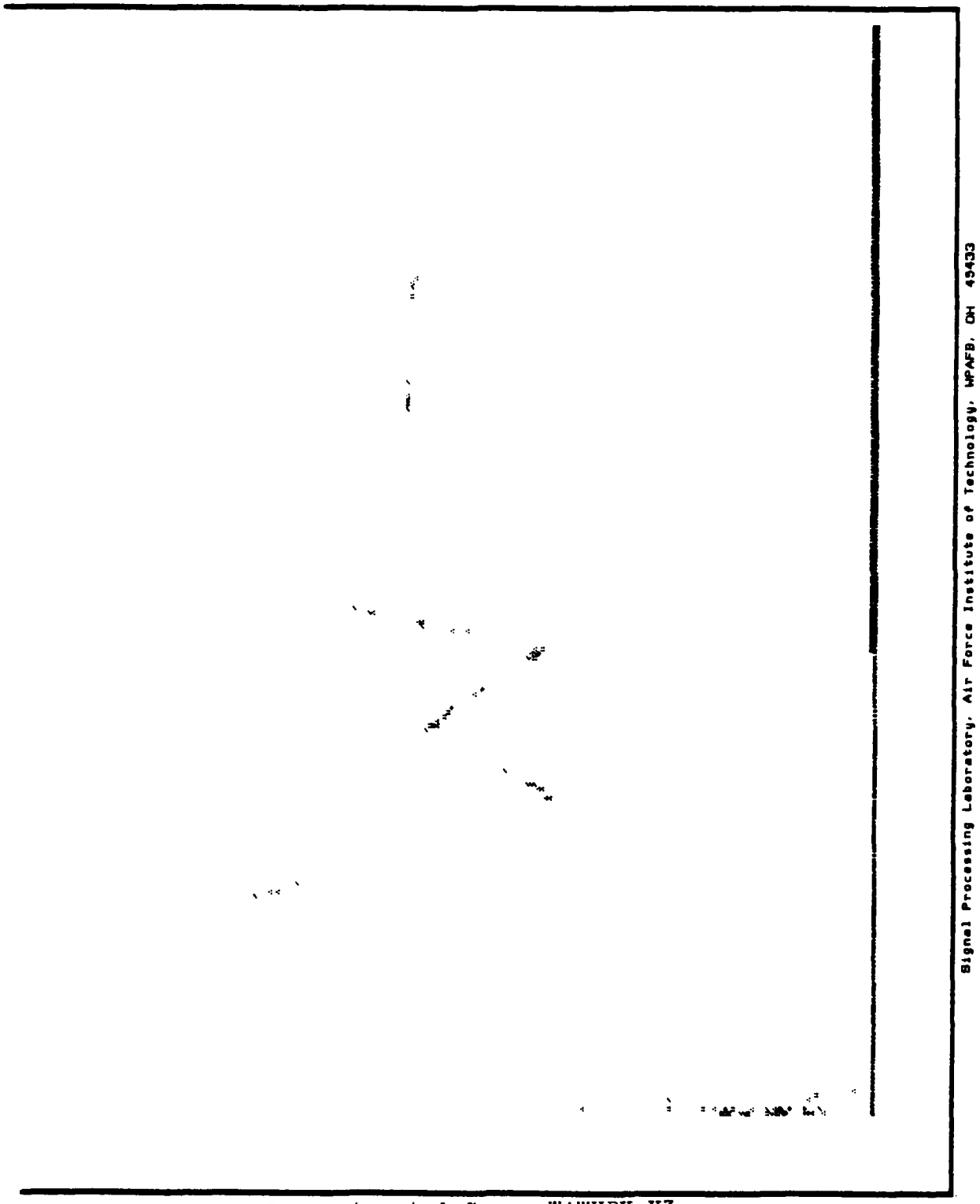
Scene WA
Figure 4.18



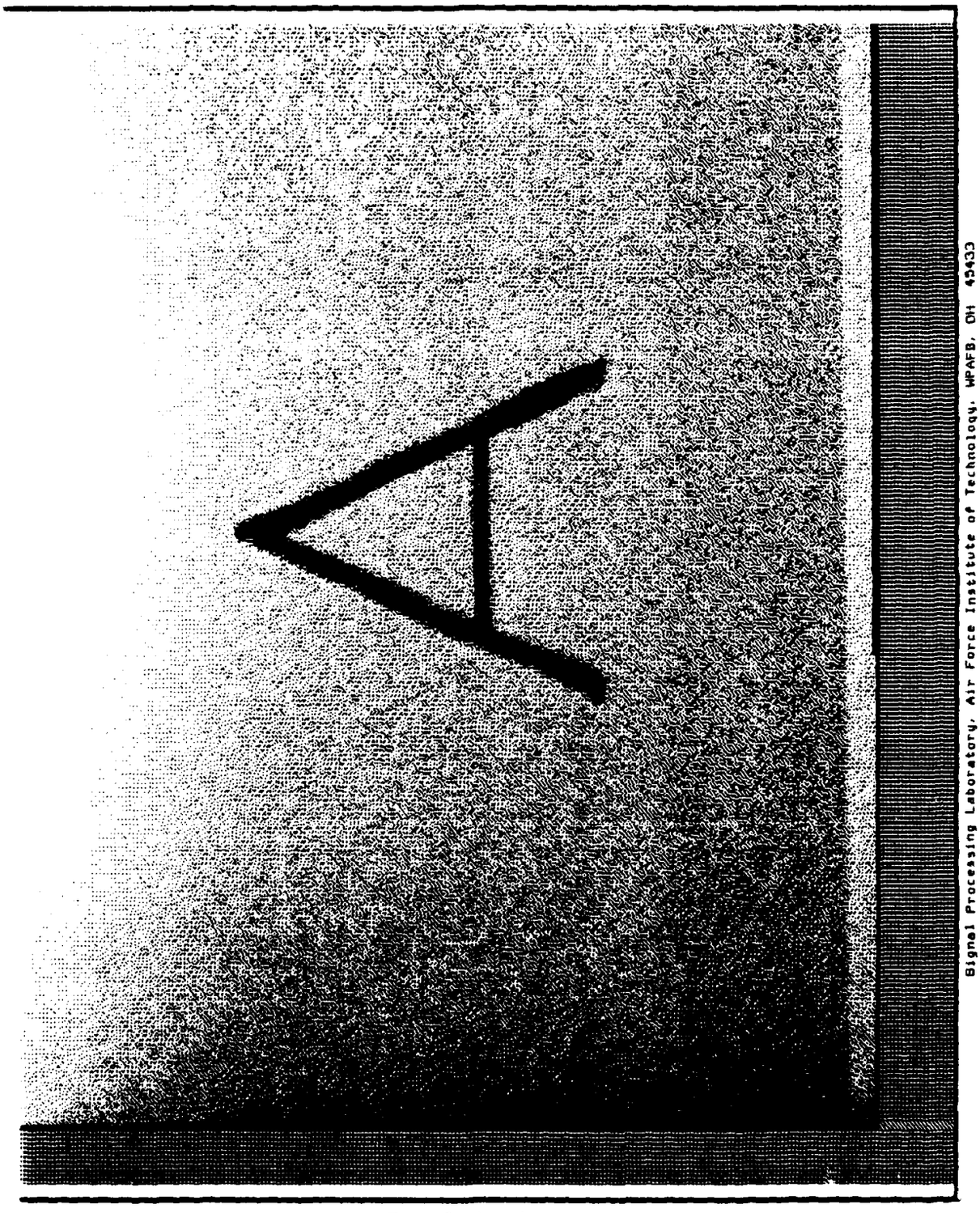
Template W
Figure 4.19



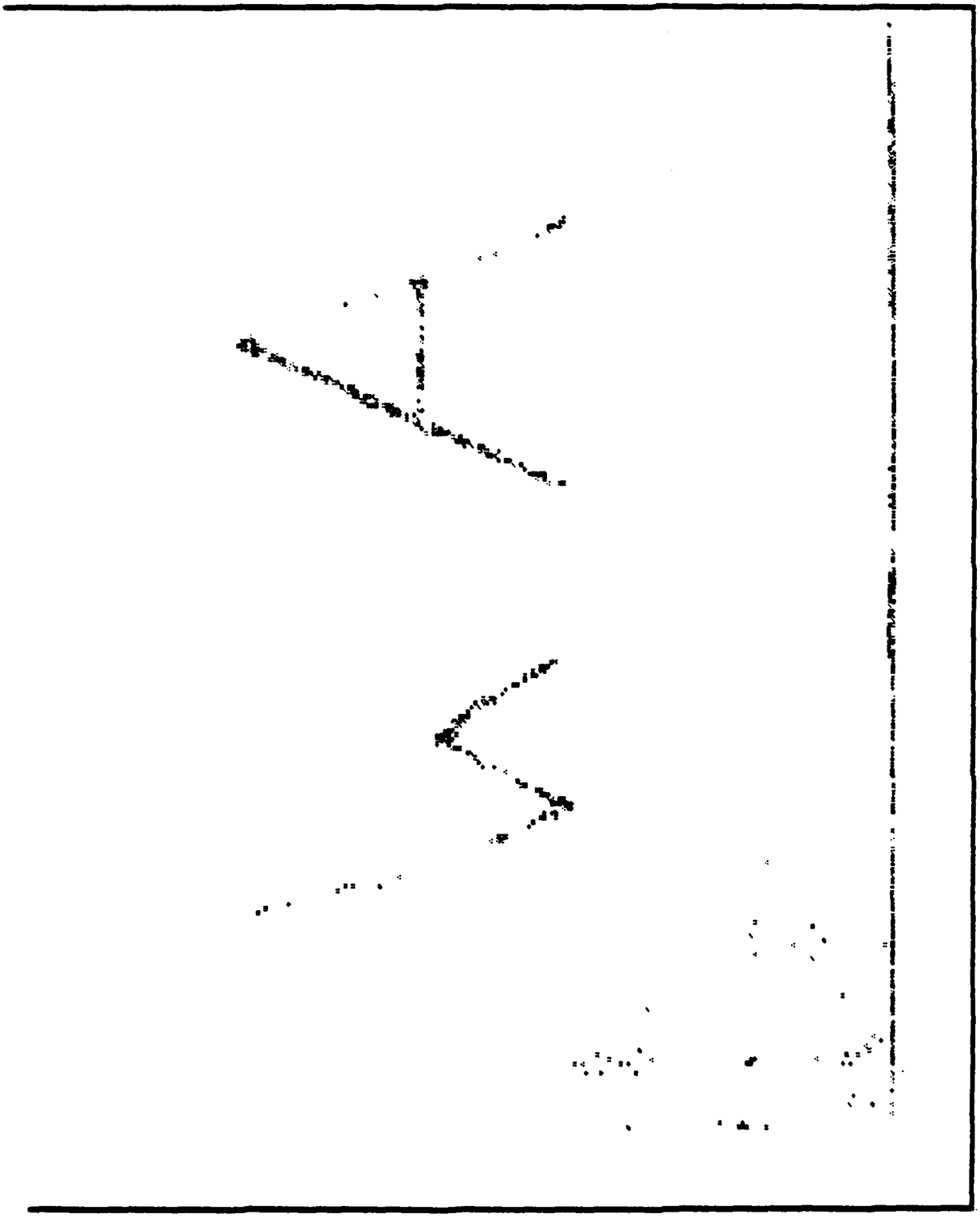
Reconstructed Scene WAWHRV.V2
Figure 4.20



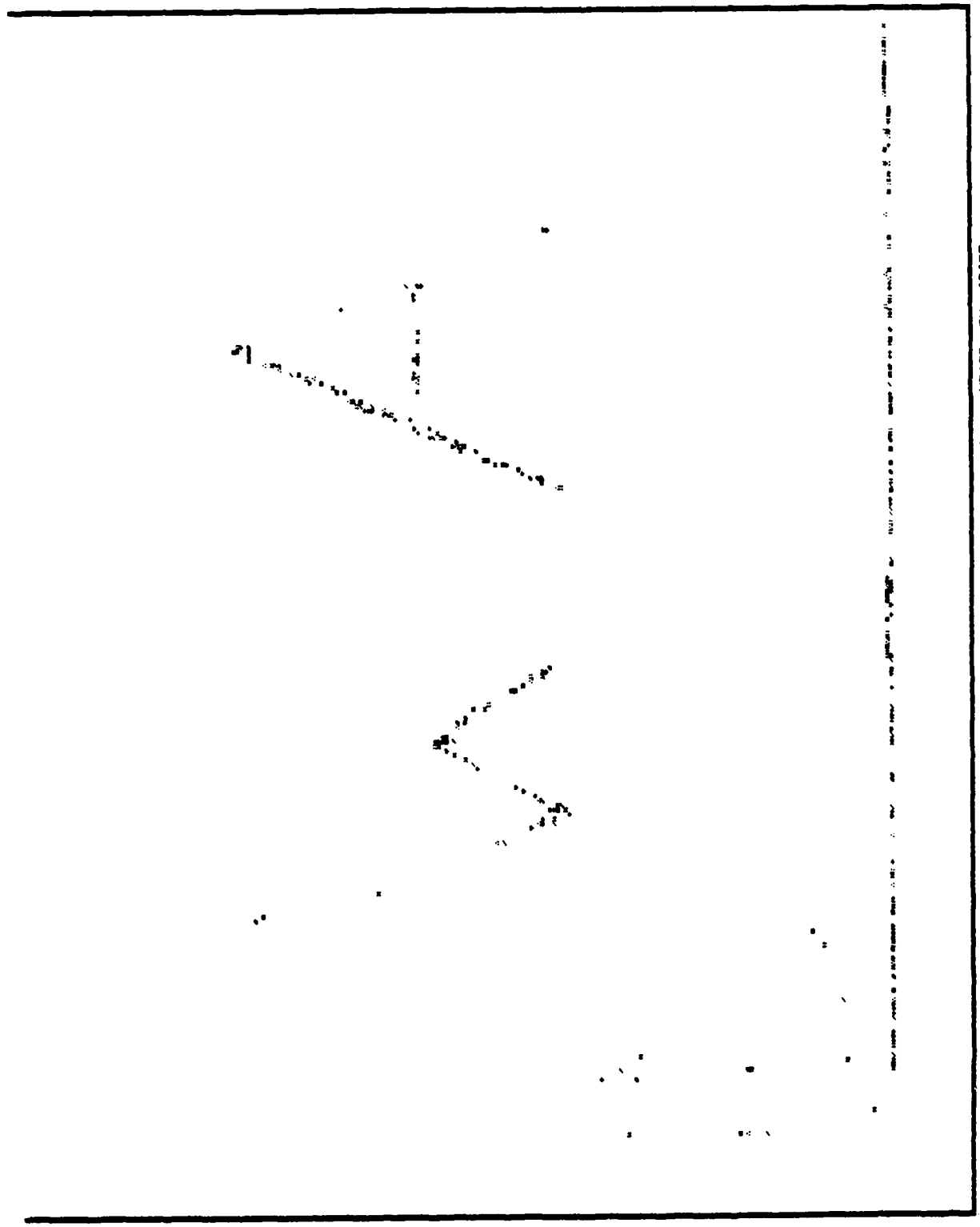
Reconstructed Scene WAWHRV.V3
Figure 4.21



Template A
Figure 4.22



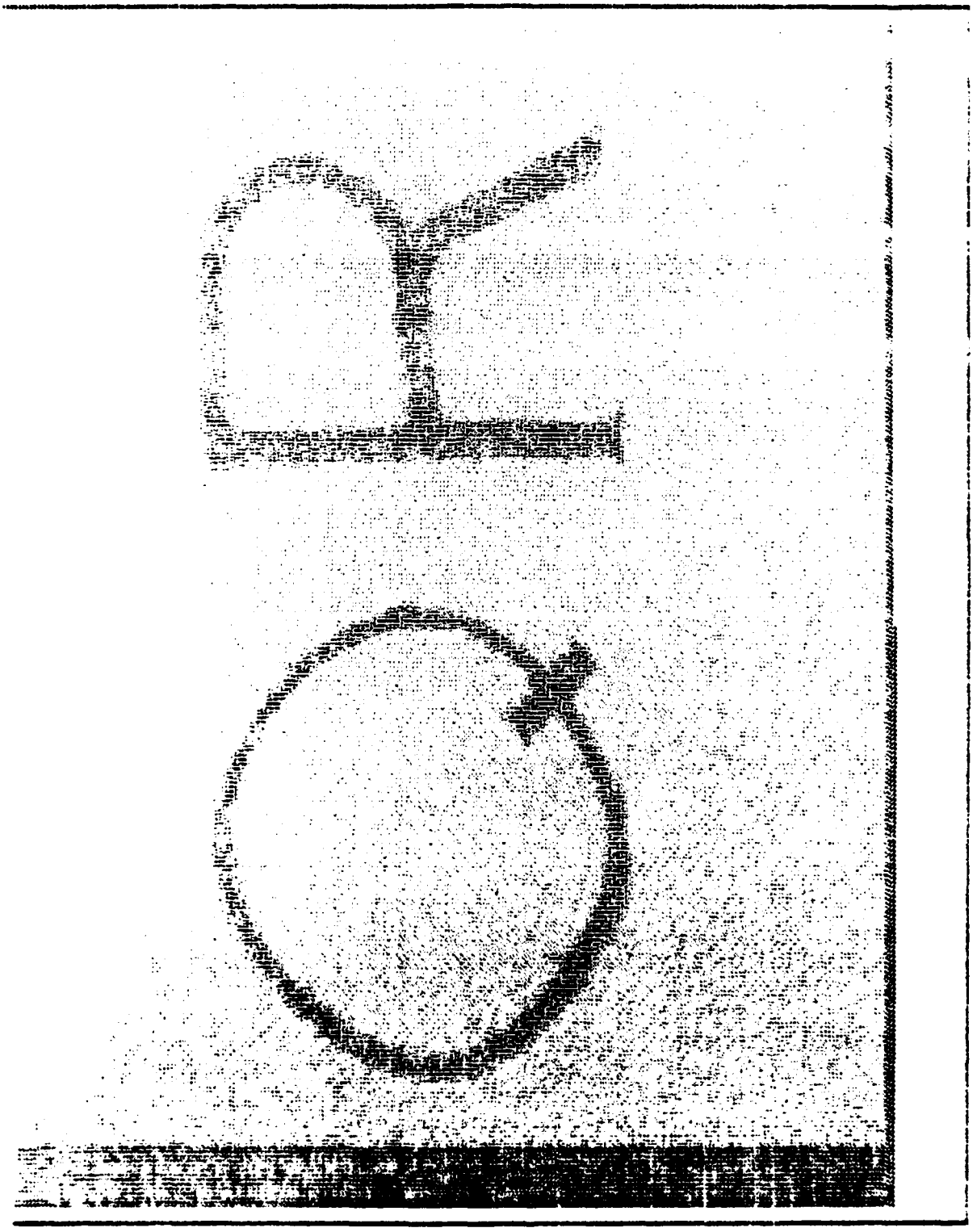
Reconstructed Scene WAAHRV.V2
Figure 4.23



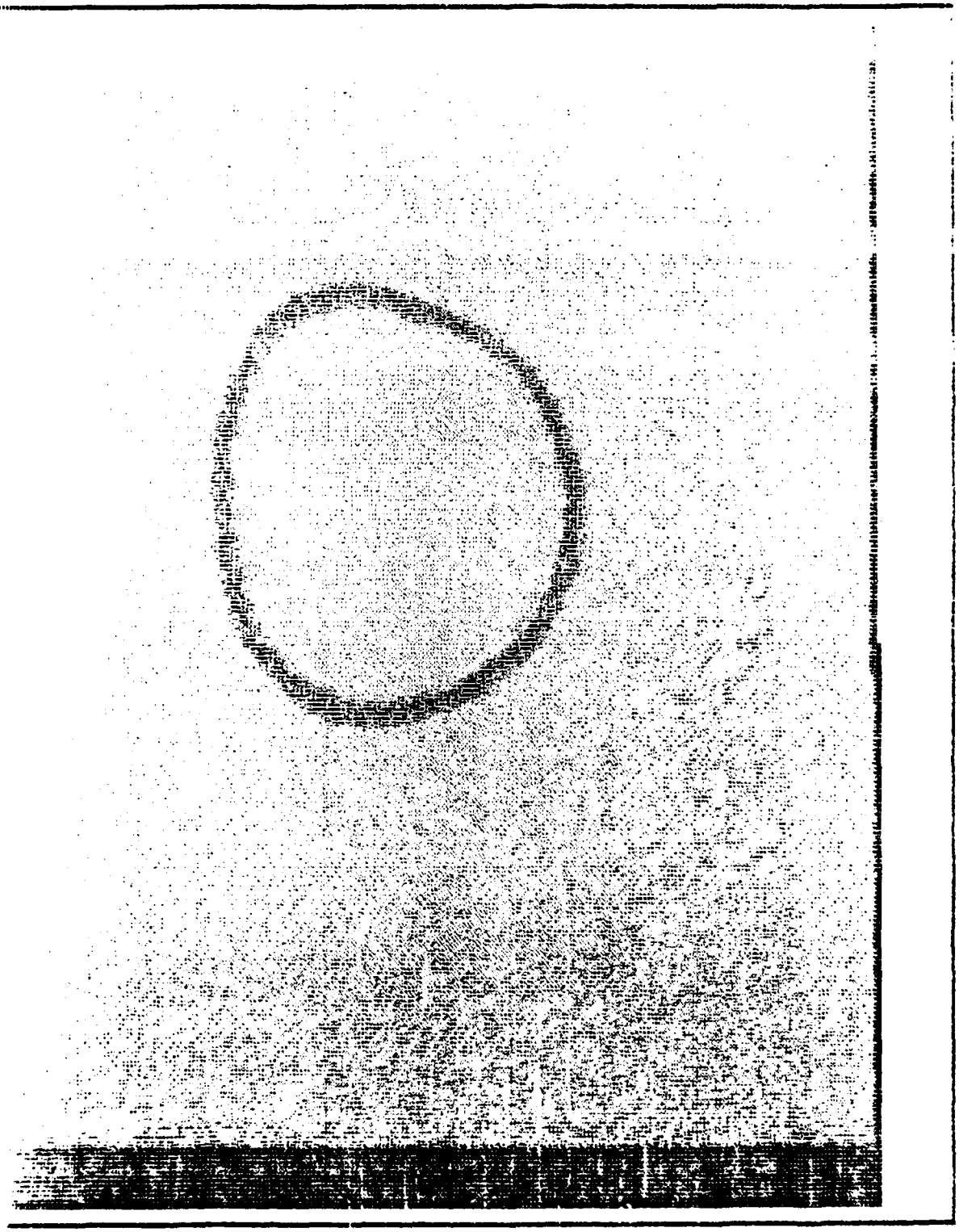
Reconstructed Scene WAAHRV.V1
Figure 4.24

The only difference between these two characters is the extra stroke in the "Q." Using "O" as a template should segment both the "O's" and "Q's"; this does happen. The correlation process that occurs after segmentation would have to choose the proper character. Figure 4.25 is of the character pair "QR." Both have "roundness" to their shapes (the top half of the "R" is reasonably round). Using "O" as the template (figure 4.26), the "Q" was segmented (figure 4.27). QROHRV.V2 has a maximum level of 0.5100, with a minimum of 0.0. A threshold could easily pick off the "Q" and not the "R." Dropping down to a lower range (figure 4.28), QROHRV.V1 has most of the "R" removed. The maximum is 0.4420, and the minimum is 0.0348. Reducing the maximum to a level similar to the levels used in the examples above (maximum: 0.3741) results in QROHRV.V3 (figure 4.29). There are no "R" pixels, while the "Q" is still present on the left and right.

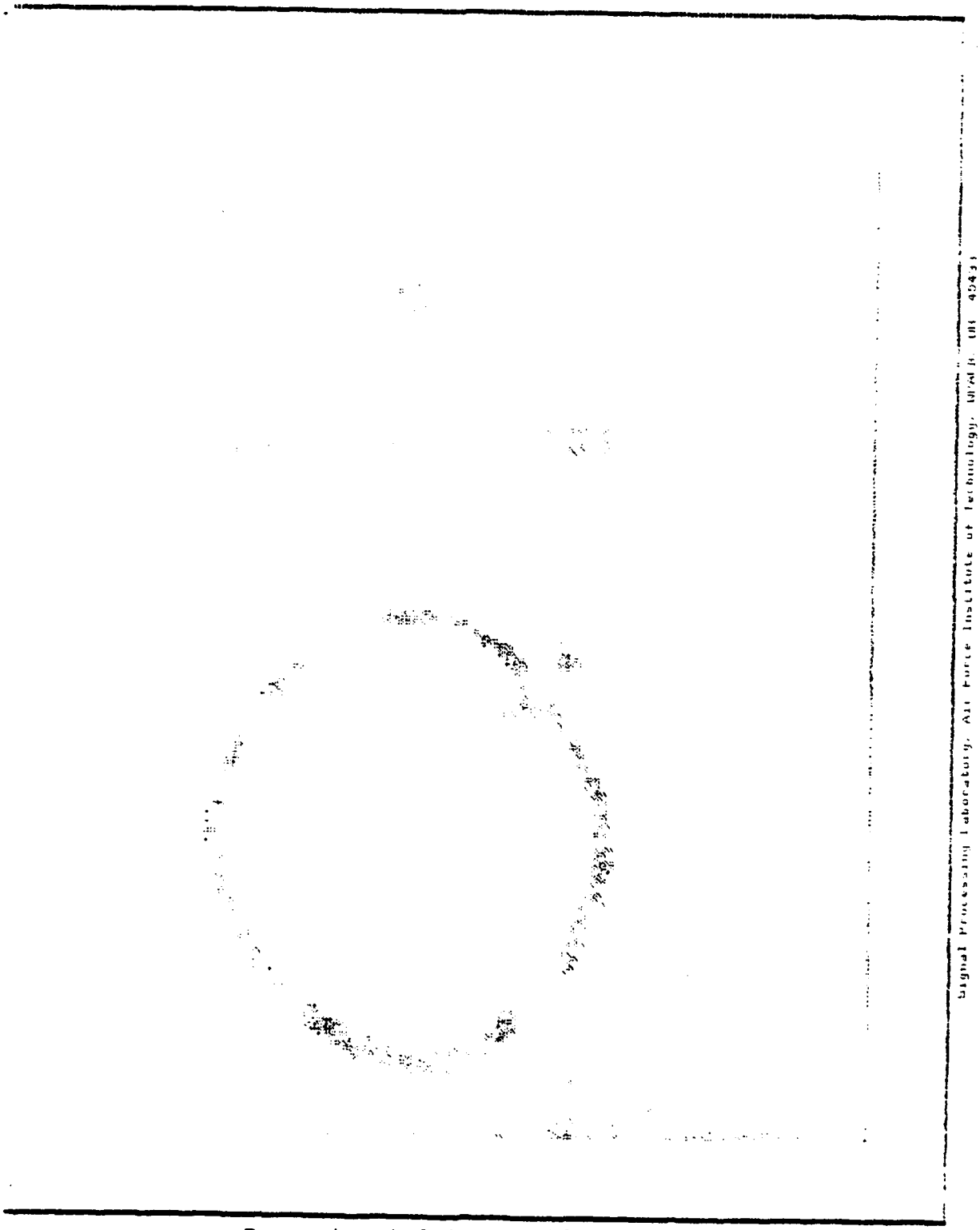
Similarly, many characters have vertical strokes (like "I") and could be highlighted by using "I" as a template. Other characters have two vertical strokes (H, M, N, and U) and using "I" as a template should pick up the vertical lines in these characters. To test this theory, two I's were used as a template (figure 4.30) to segment the "U" from the "S" in the scene "US" (figure 4.31). USIIHRV.V1 has a maximum magnitude of 0.5973 and a minimum of 0.0 (figure 4.32). Most of the "S" is gone, while the "U" is clearly visible. A threshold should be able to pick off the higher level of pixels that are present in the "U." However, dropping to a



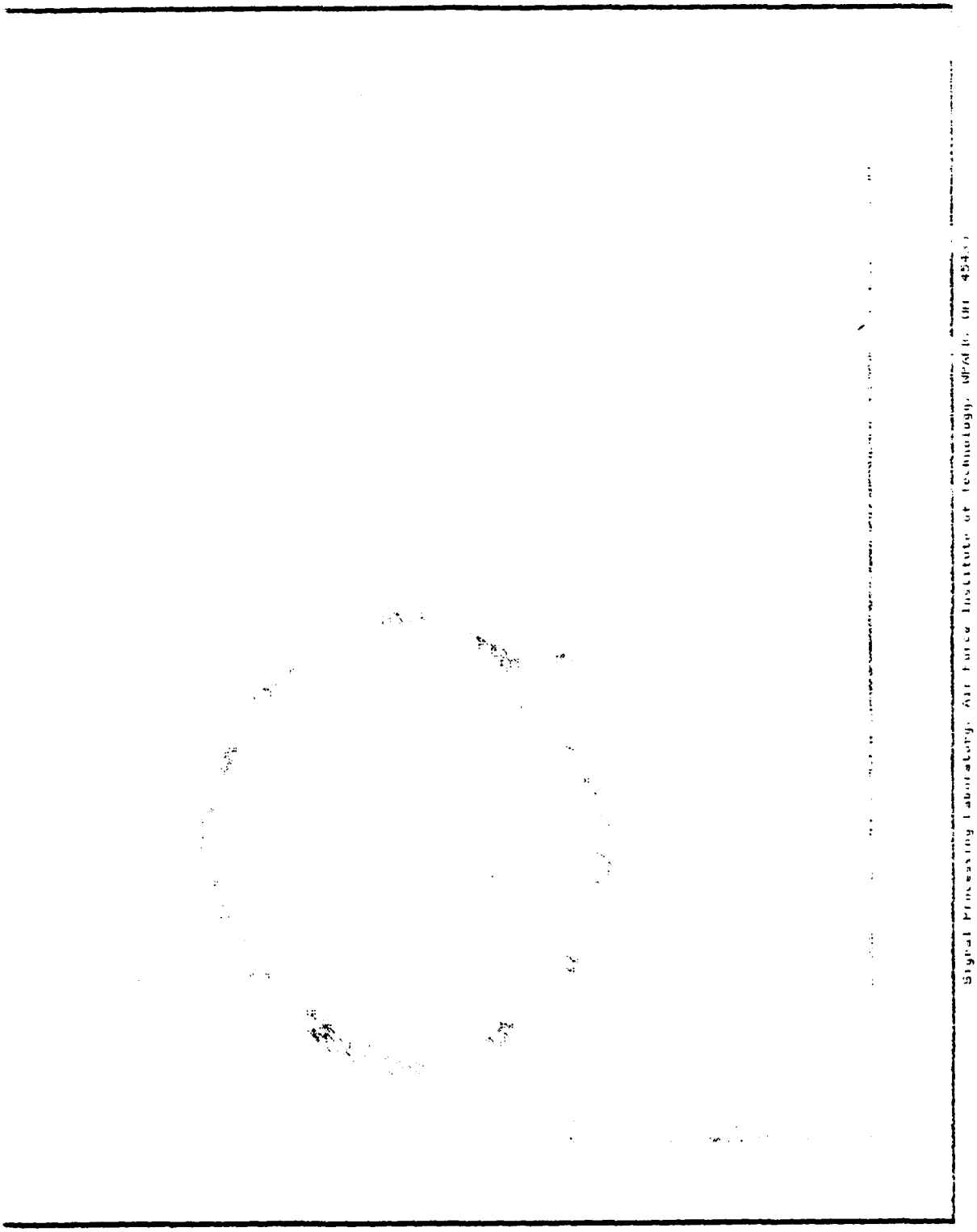
Scene QR
Figure 4.25



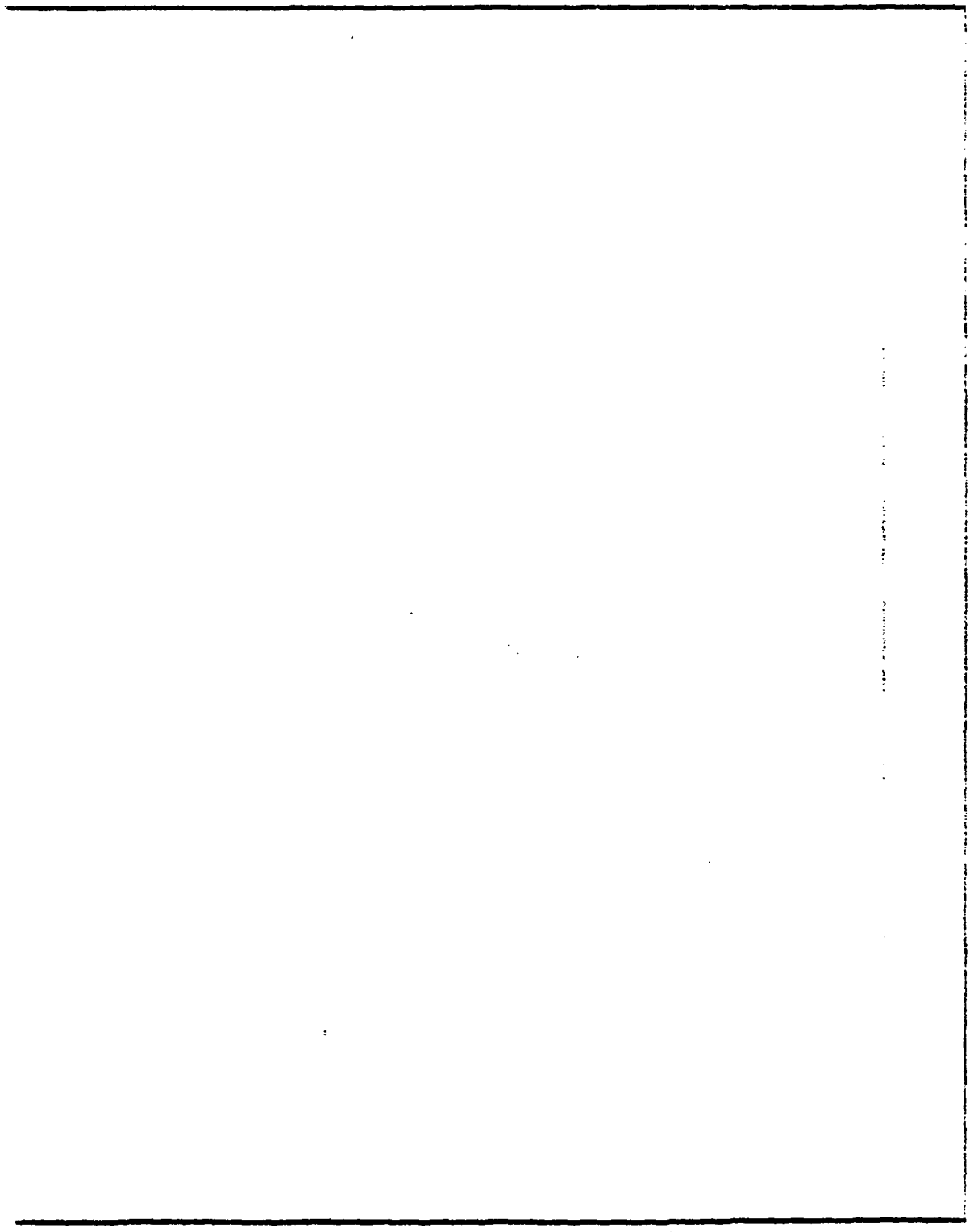
Template 0
Figure 4.26



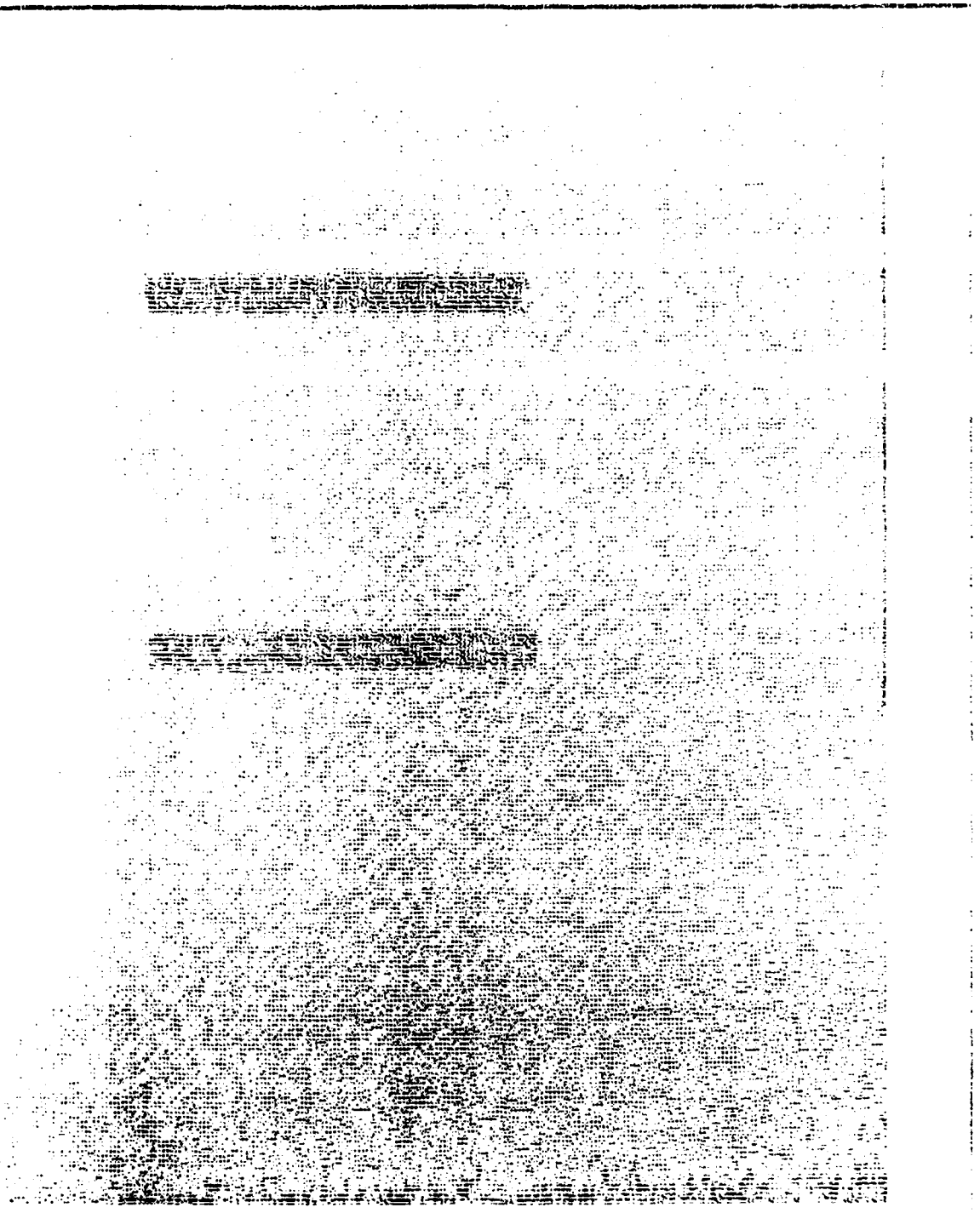
Reconstructed Scene QROHRV.V2
Figure 4.27



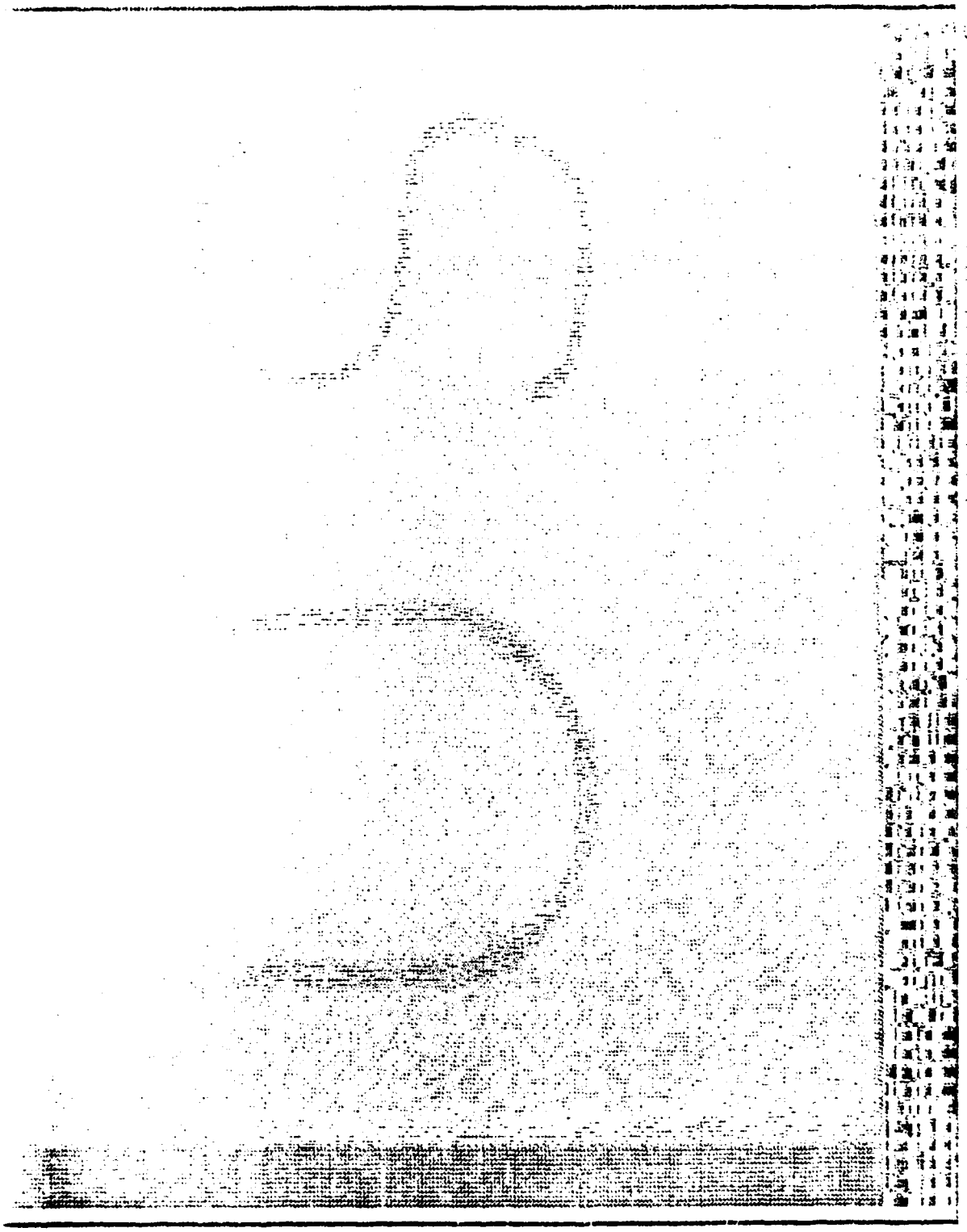
Reconstructed Scene QROHRV.V1
Figure 4.28



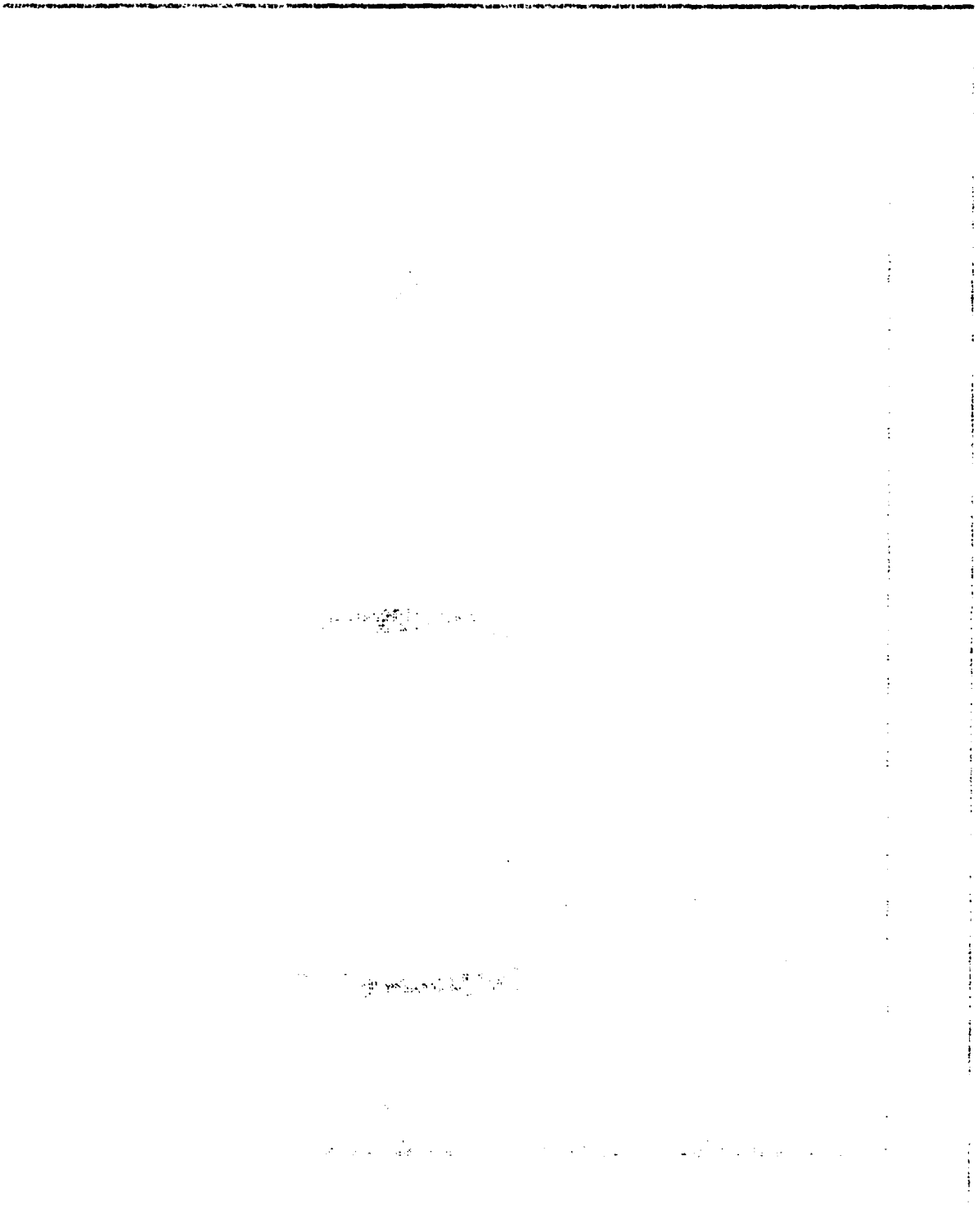
Reconstructed Scene QROHRV.V3
Figure 4.29



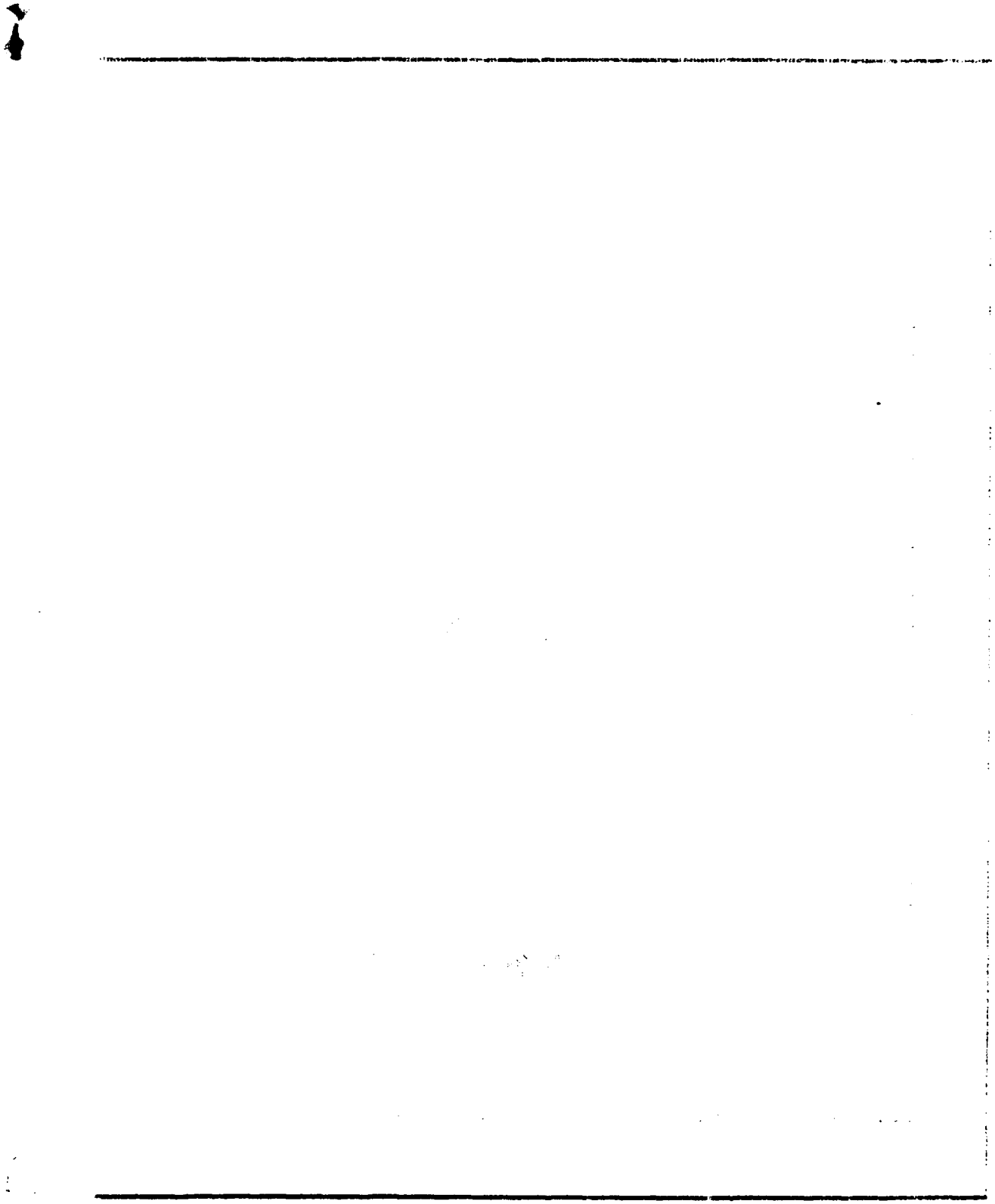
Template II
Figure 4.30



Scene US
Figure 4.31



Reconstructed Scene 4.32
Figure 4.32



Reconstructed Scene USIIHRV.V3
Figure 4.33

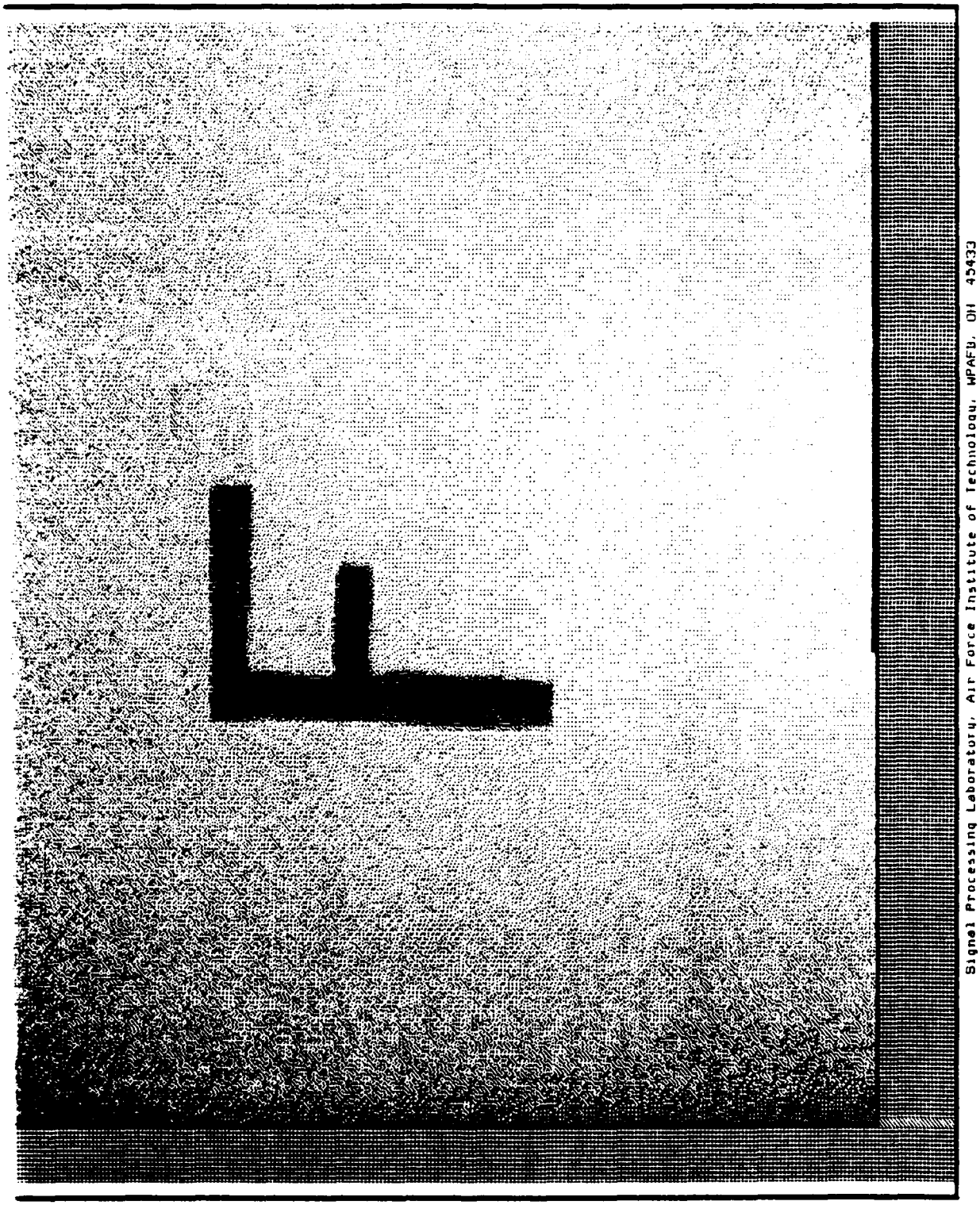
lower maximum magnitude level in USIIHRV.V3 (figure 4.33), with the maximum level at 0.5347, we see that the level of the "U" is far above the level of the "S." From these last two examples, we can see that care must be used in selecting the template characters.

The next two sets of characters presented both have the characters touching each other or overlapping vertically. This is where segmentation is most difficult with methods that attempt to segment characters by finding gaps between them; these methods cannot find gaps between touching characters.

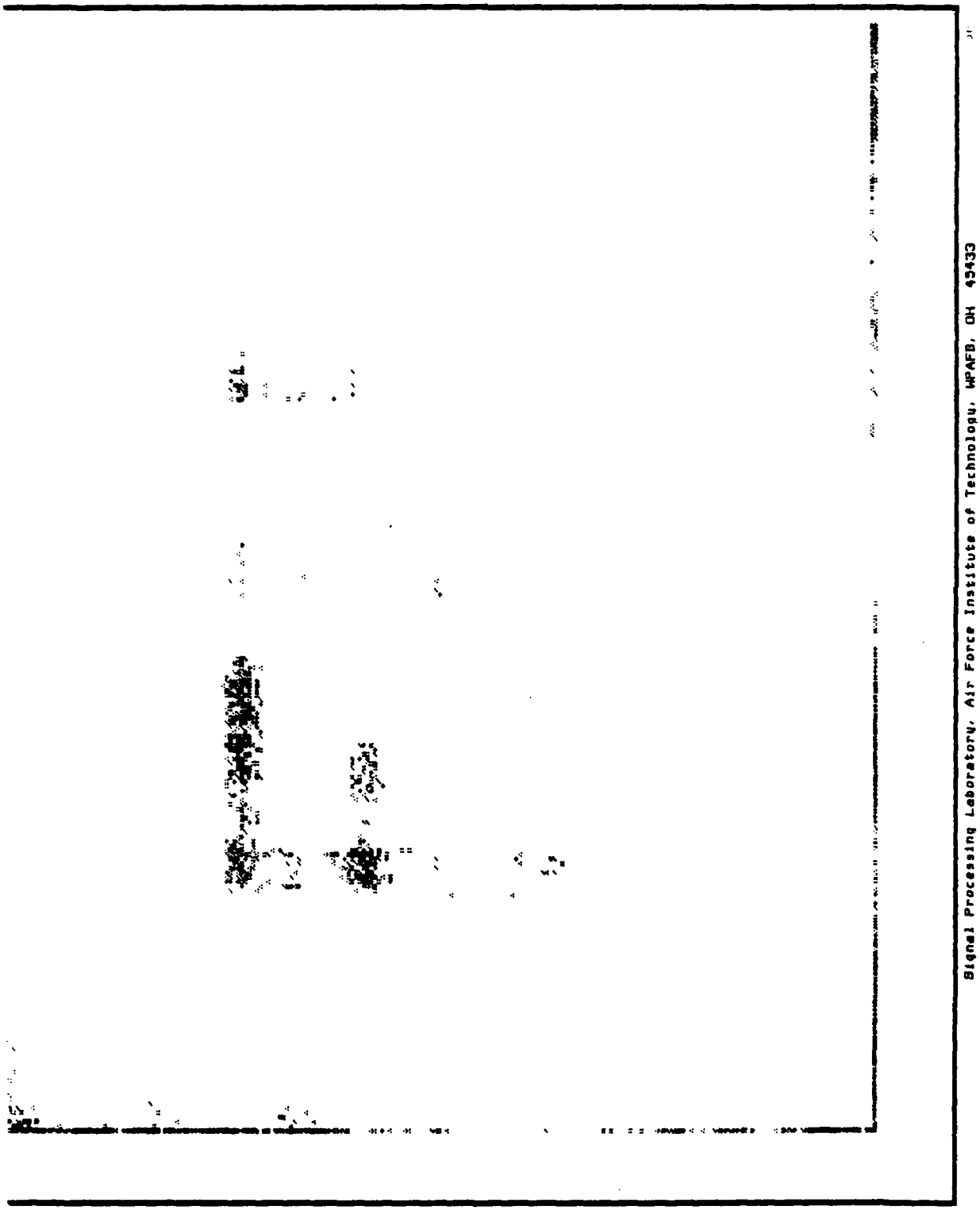
The first set is the three-character group "FAD" (figure 4.34). The "F" touches the "A" and the "A" touches the "D." The template character is "F" (figure 4.35). FADFHRV.V3 has a maximum of 0.2710 and a minimum of 0.0 (figure 4.36). The "F" can easily be segmented by thresholding. Dropping the maximum level to 0.1450 in FADFHRV.V2 (figure 4.37), the "F" is still quite visible, and its width would be easily picked up by thresholding.

The second set of harder-to-segment characters is the three-letter group "GYJ" (figure 4.38). The "G" touches the "Y" and the "Y" and "J" overlap vertically. The template character is "Y" (figure 4.39). GYJYHRV.V6 is shown in figure 4.40. Its maximum is 0.4070, and its minimum is 0.0679. The "Y" clearly has darker pixel levels than the "G"; dropping the maximum to 0.3400 and the minimum to 0.0 results in GYJYHRV.V5 (figure 4.41). The "G" is virtually

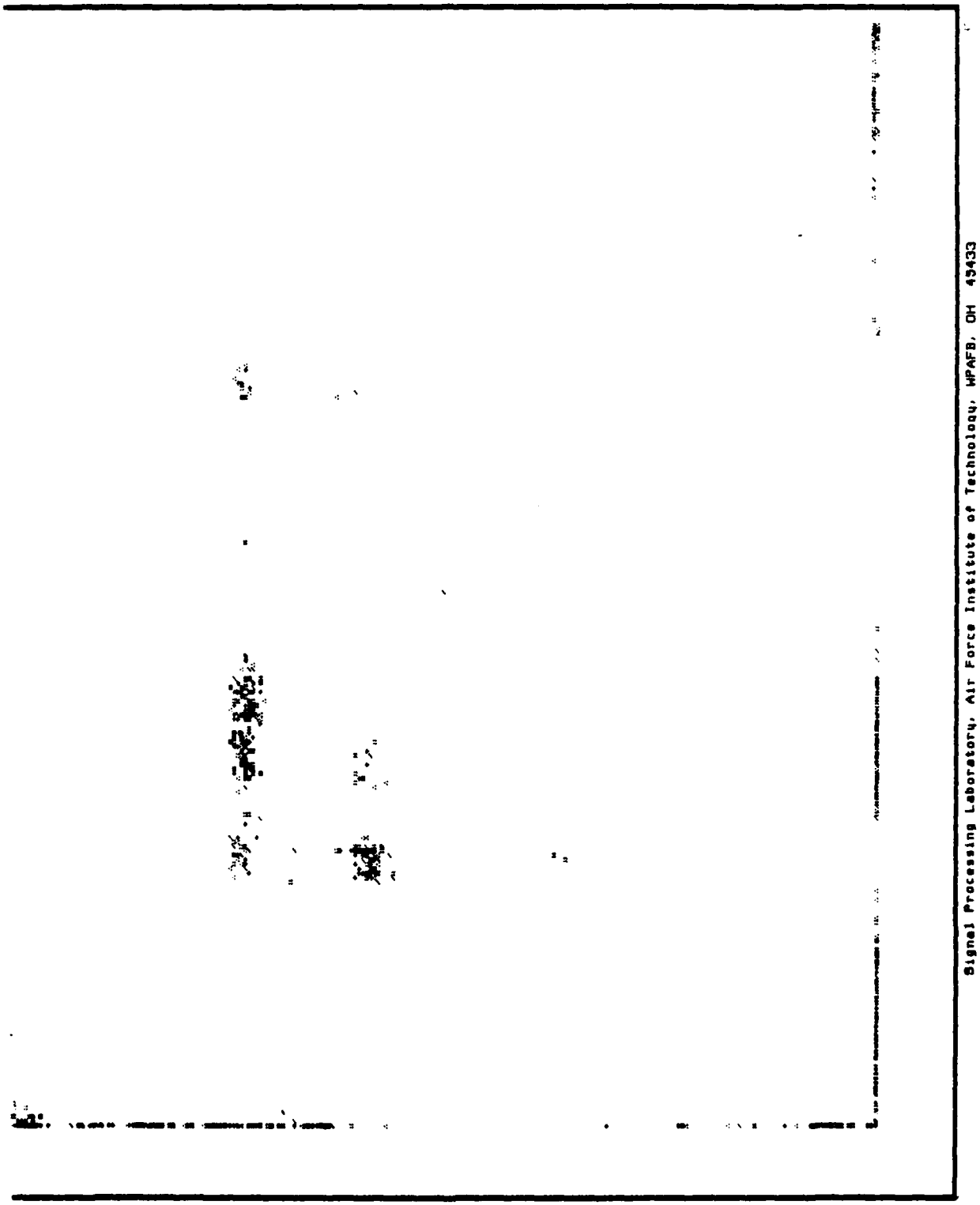
Scene FAD
Figure 4.34



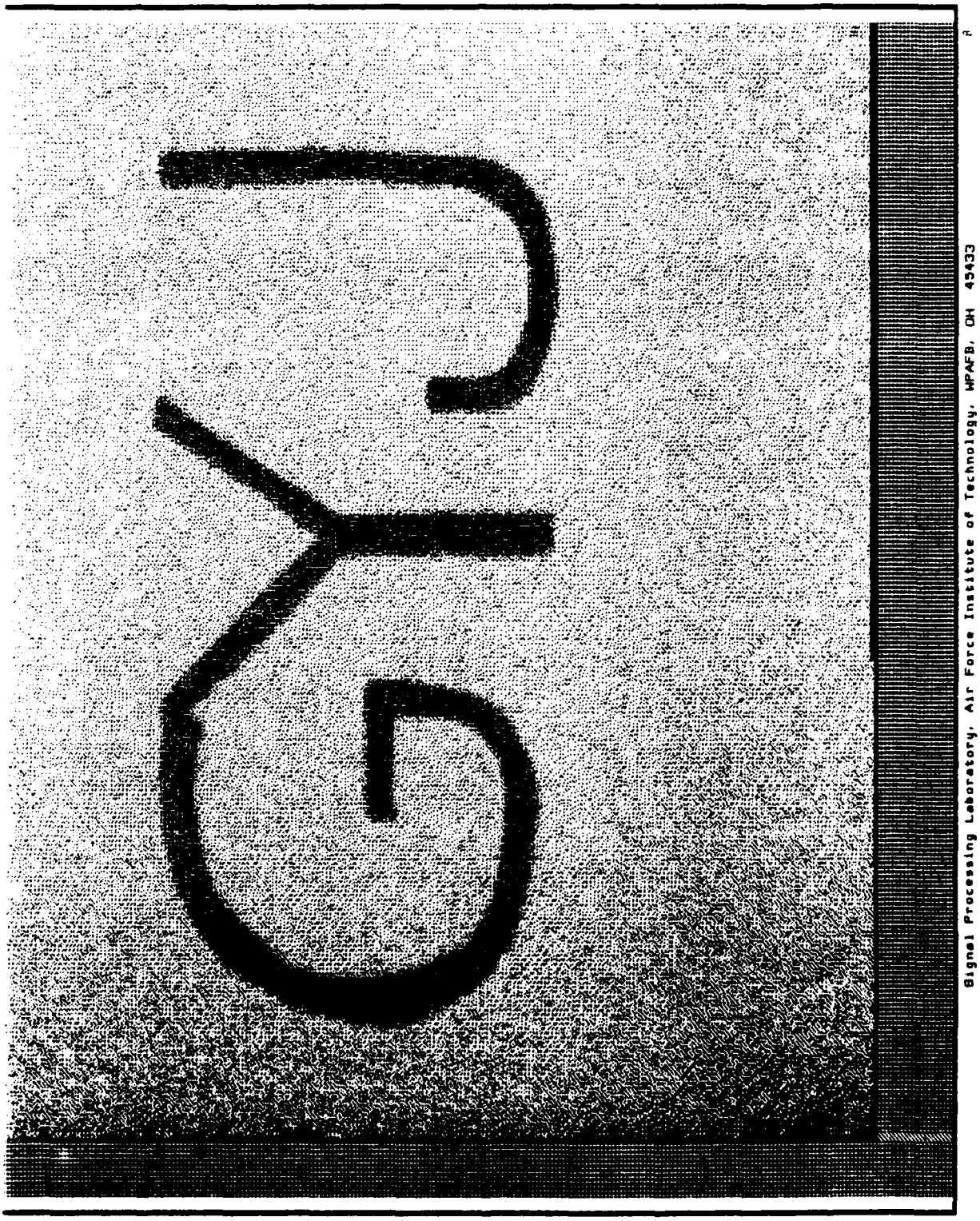
Template F
Figure 4.35



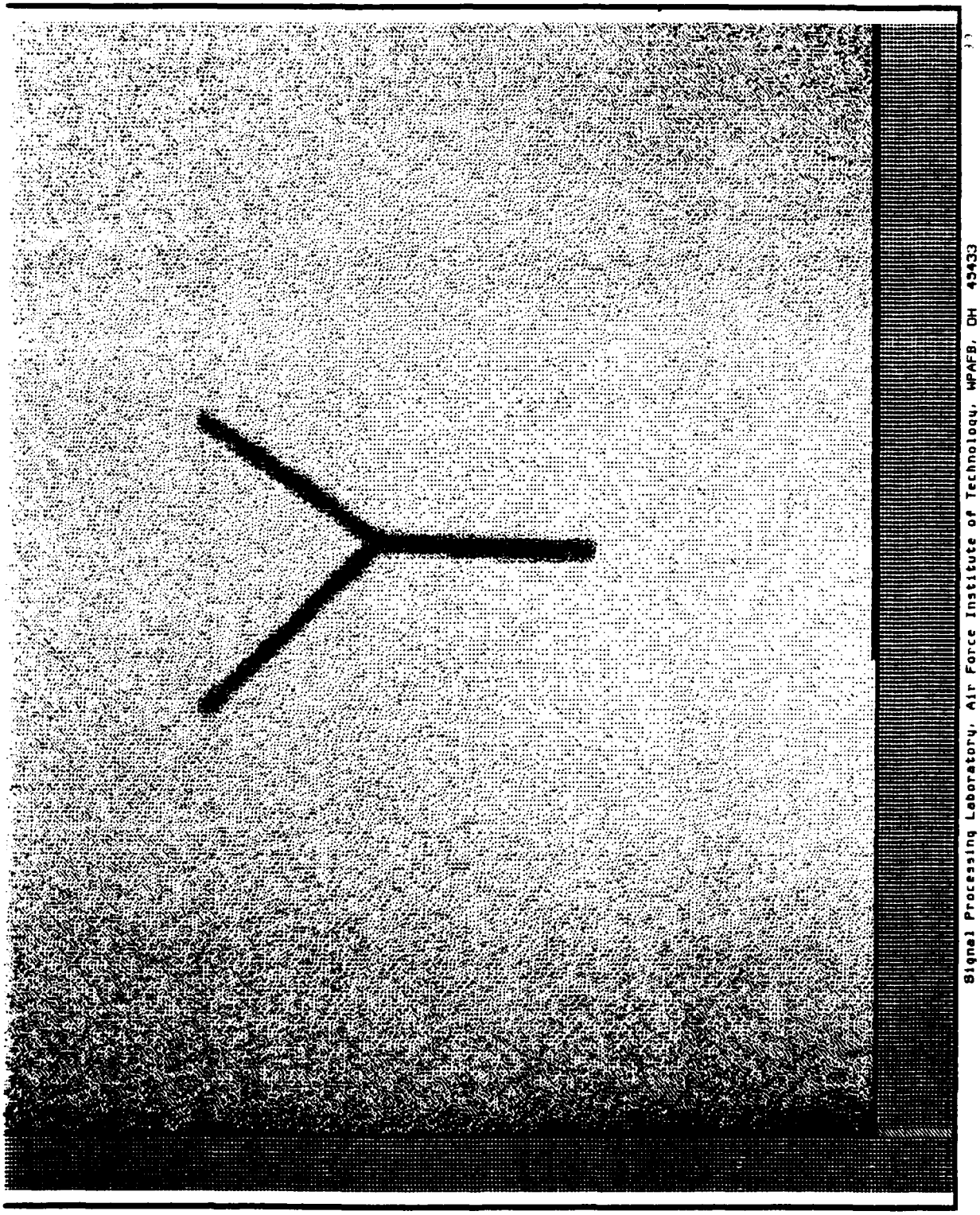
Reconstructed Scene FADFHRV.V3
Figure 4.36



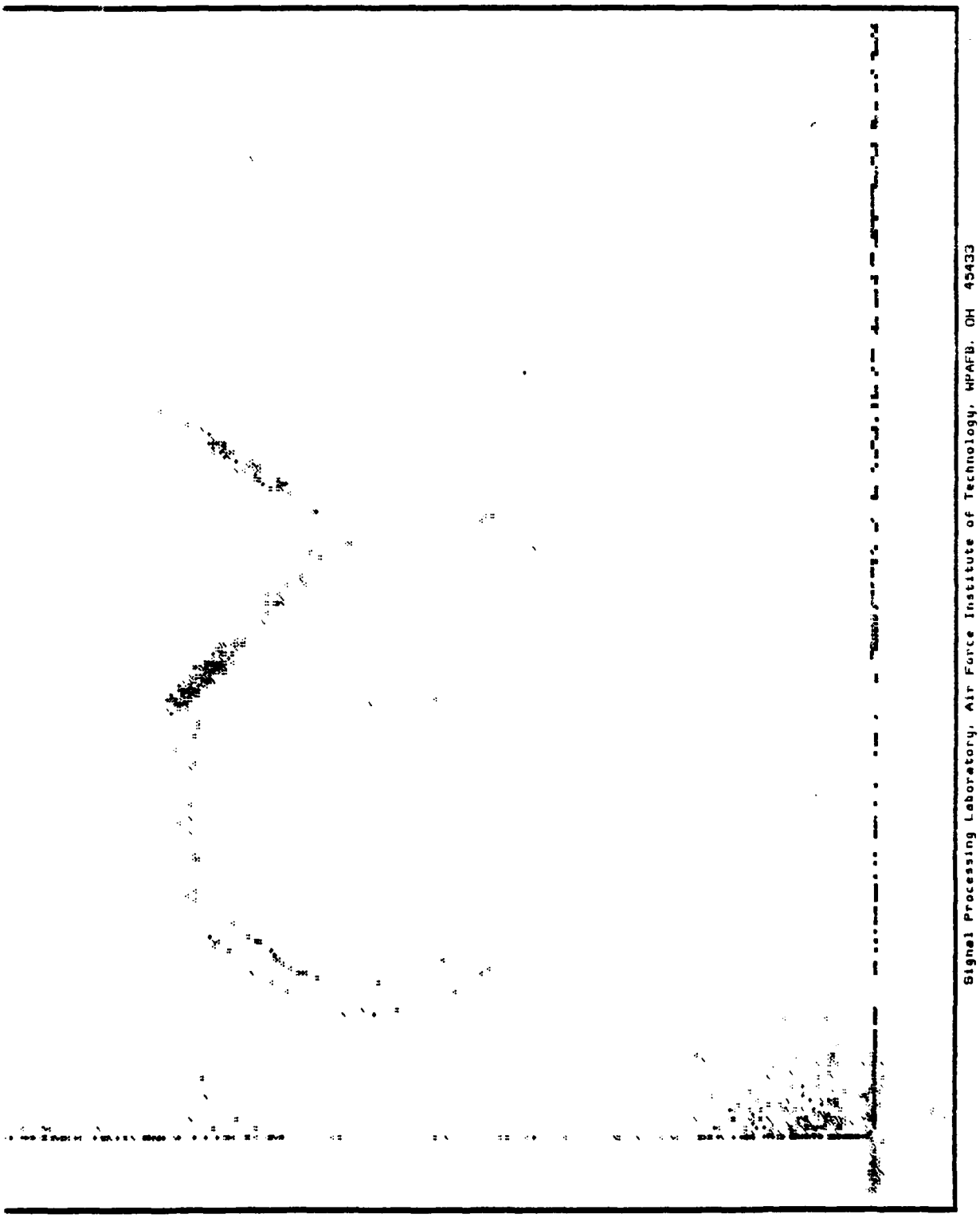
Reconstructed Scene FADFHRV.V2
Figure 4.37



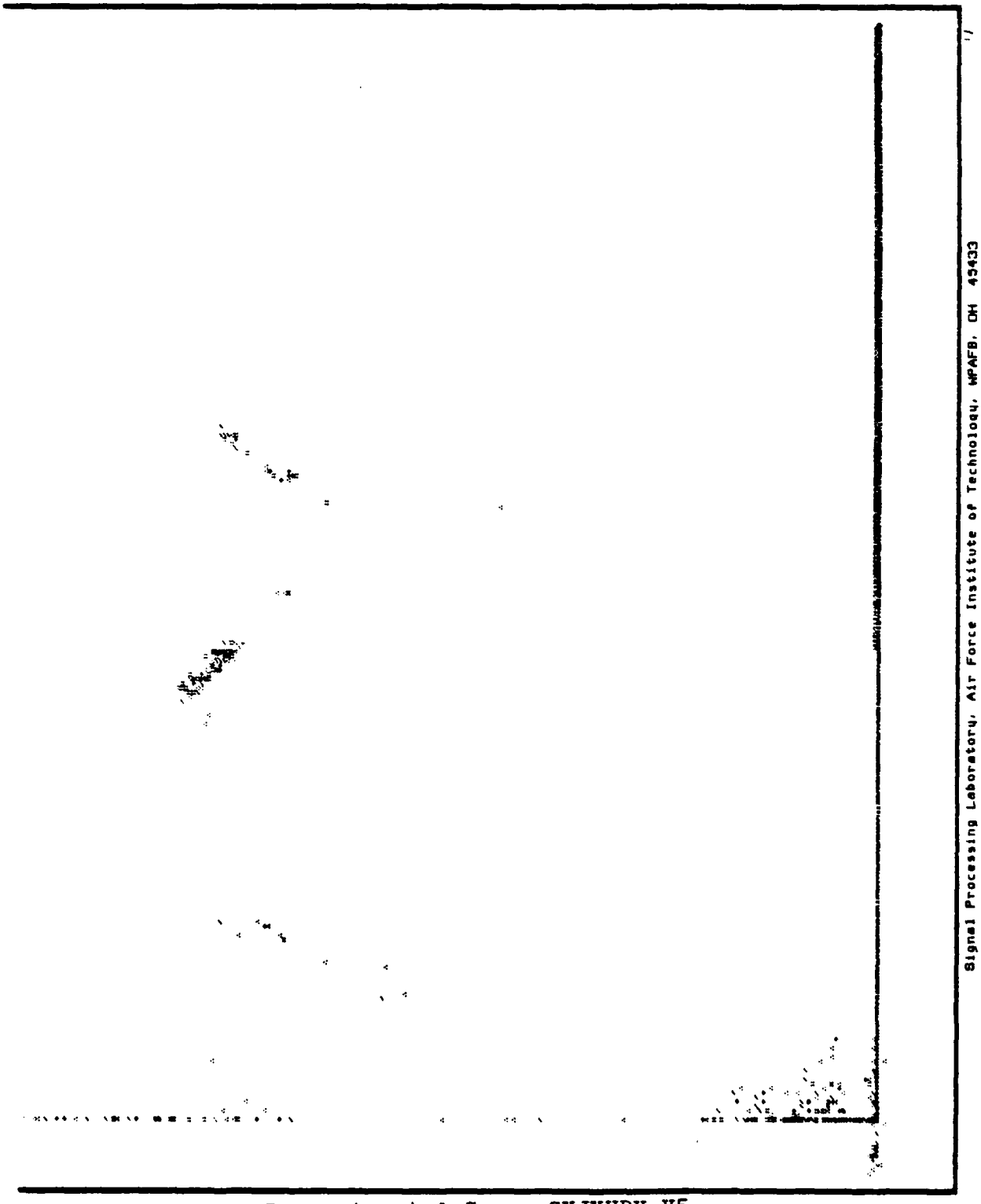
Scene GYJ
Figure 4.38



Template Y
Figure 4.39



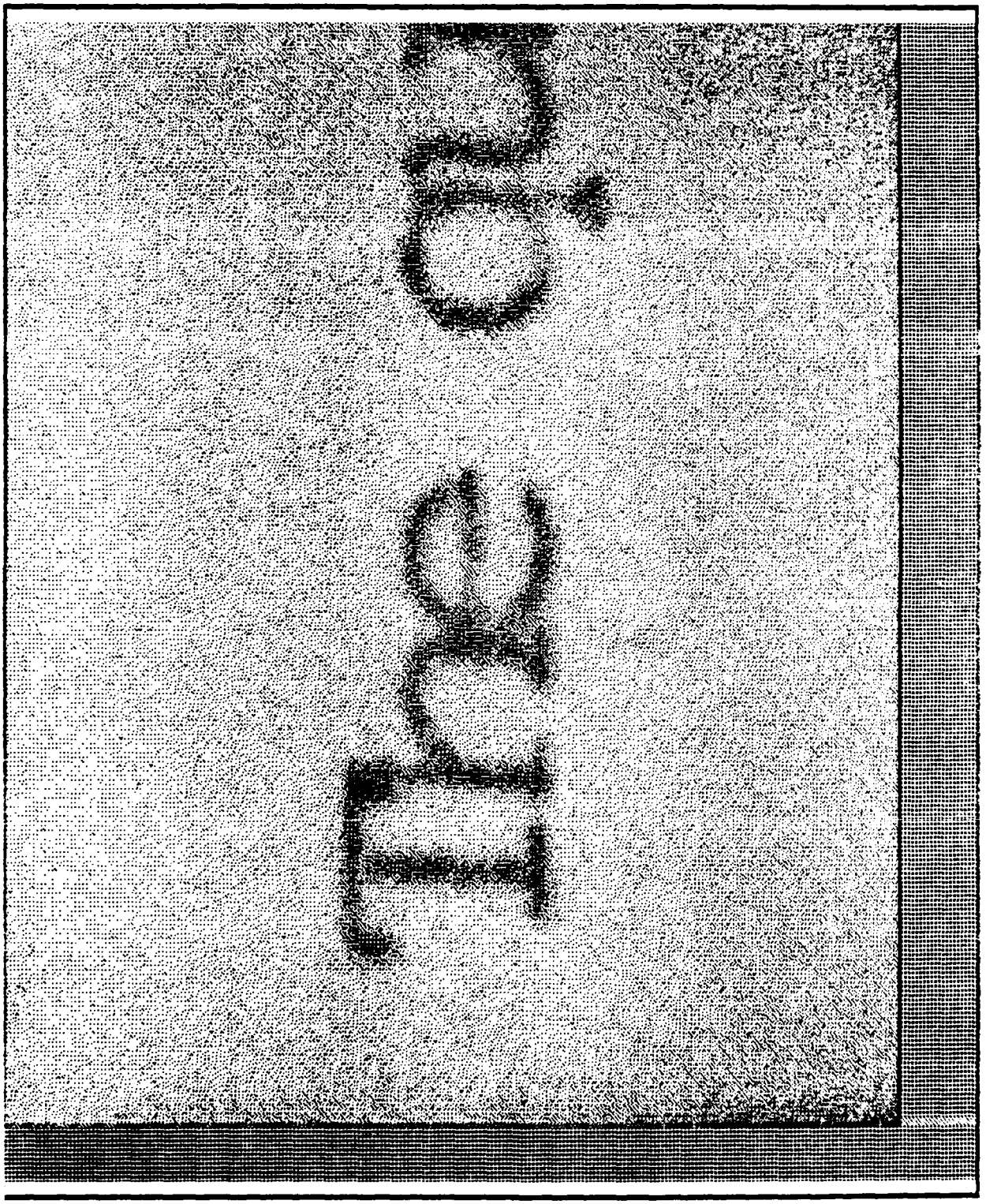
Reconstructed Scene GYJYHRV.V6
Figure 4.40



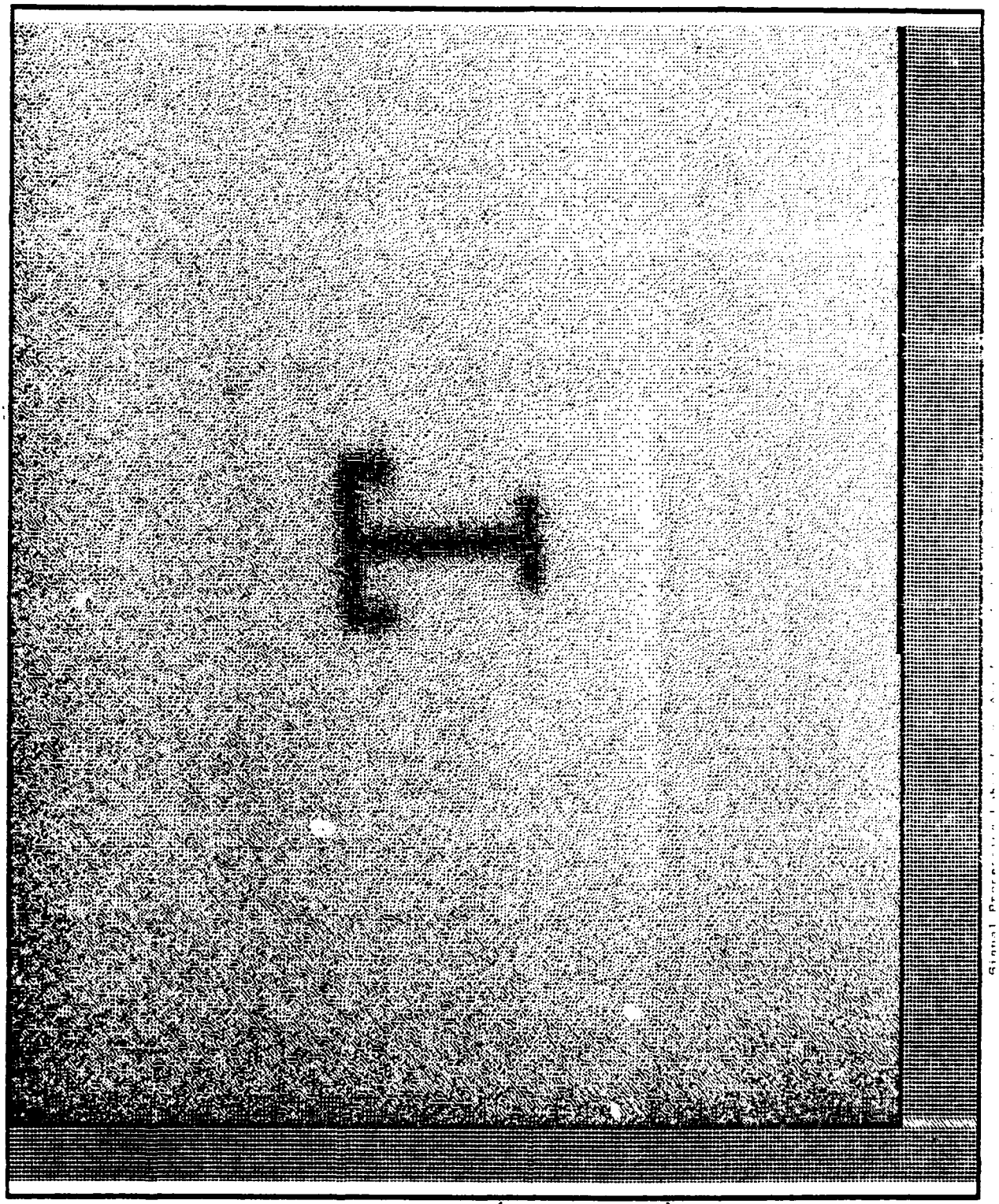
Reconstructed Scene GYJYHRV.V5
Figure 4.41

removed, and the "Y" can be picked up by a threshold.

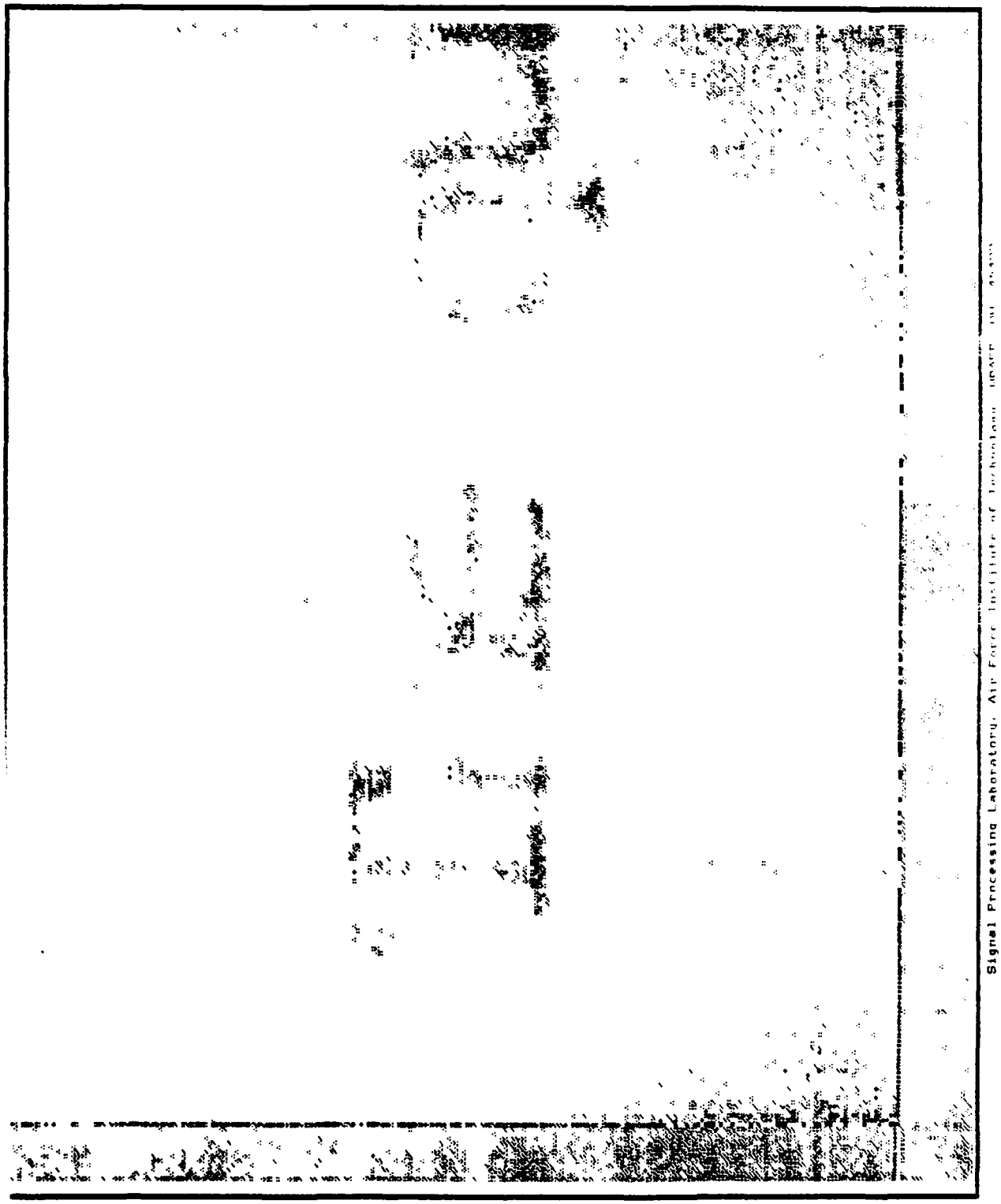
In the examples included in this chapter, large, handwritten characters were used. With this algorithm, smaller characters were not segmentable; when smaller characters were used, the original scene was recreated in virtually every attempt. For example, the typewritten scene "The quick" (figure 4.42) was used to test the algorithm; the scene has a low character size-to-window size ratio. The template was "T" (figure 4.43). The segmentation scene, THE8THRV.V9 (figure 4.44), shows that most of the characters are still present; a window formed by thresholding would not reduce the amount of characters in the correlation process. This occurred because the phase information contained the whole scene. With the larger characters, however, excellent results occurred. Only with the set "WA" was there any problem. This shows why a two-step process is necessary for unformatted character recognition: the segmentation process will only bound the most likely location of a character. A correlation process will be needed to insure that the character is properly recognized.



Scene "The quick"
Figure 4.42



Template T (typewritten)
Figure 4.43



Reconstructed Scene THE8THRV.V9
Figure 4.44

V. Conclusions and Recommendations

Review.

Chapter two developed the theory behind this segmentation algorithm. Chapter three described the software that was developed to perform the reconstruction algorithm, and detailed listings are included in Appendix. Chapter four presented the segmentation results for half of the English uppercase alphabet. Two conclusions can be reached, and I have three recommendations.

Conclusions.

A simplified version of Horev's algorithm shows great promise in segmenting characters. Since characters are usually not rotated, and their height can be set to a constant level by an optical process before segmentation, the segmentation algorithm only has to find the left-to-right range of each character. Although researchers [11] have shown that Horev's algorithm has certain difficulties performing its task (locating targets in a cluttered scene independent of size, location, or rotation), it performs well in the special case of characters on a contrasting background.

Regardless of this algorithm's performance as shown in chapter four, there is a limitation. The algorithm does not perform well on "small" characters (in comparison to the

2D-DFT window size). Either the characters have to be made larger (optically or by software), or the window has to be made smaller (a software change). Either or both of these changes could conceivably make this algorithm work for most character sizes.

Recommendations.

Three recommendations can be made. They are: (1) Software has to be written to reduce the window size for the 2D-DFT; (2) An algorithm has to be written to perform the threshold operation and place a "boundary" around the segmented character(s); and (3) The effect of noise on the algorithm has to be studied.

A program can be written to place a window around any location in the scene. A 256 by 256 window is used in this algorithm; a 64 by 64 window should work well for very small characters (e. g. characters from printed material). This will also reduce the 2D-DFT runtime by a large factor. Segmentation of smaller characters can be studied if this type of program is written. The main reason for a software change, and not changing the video digitizer to a smaller size, is speed. The hardware can digitize 256 by 256 pixels practically as fast as 64 by 64 pixels; if the window program is developed, it can partition the 256 by 256 pixel scene into 16 subscenes very rapidly, instead of requiring 16 separate input operations.

An algorithm to place a boundary around the segmented

character has to be able to include the proper pixel magnitude range that represents the character that should be segmented. More research is needed to develop this general threshold criterion. Table I lists the magnitudes included as pixels in the scenes in this thesis. Since all the parameters (except what each character is) are held constant (height of the characters, lighting, and input device), the segmented character will always be in the lowest one-fifth of the range of magnitudes in the reconstructed scene (See Table I). Also, a minimum number of magnitudes (or pixels) are always needed to make up even the smallest character. A threshold algorithm would have to consider these characteristics, and possibly more, if it is to accurately segment characters without human intervention. Once the algorithm is developed, the segmented characters are ready to be read by a correlation process [12], and there is already a first choice for each character: the template.

The noise was not removed from any scene (except for the noise blanked with the progam VTOC; see chapter 3, page 14) during this algorithm. It was left in to check its effect on the algorithm. There are primarily two types of noise that would be included in a video scene. The first kind of noise is from the digitization process. For example, if the characters are printed on certain colors of paper, there may not be a clear distinction between the background and the characters. (The scenes in this thesis are black on white, and the digitizer rarely digitized the white background as

Table I

Maximum and Minimum Magnitude Ranges Included in each Scene by Program PICK

Segmentation Scene	Inclusion Magnitude Maximum	Minimum
KZZHRV.V4	0.2910	0.0000
KZZHRV.V3	0.2190	0.0000
KZZHRV.V2	0.1460	0.0000
KZKHRV.V3	0.2120	0.0000
KZKHRV.V2	0.1410	0.0000
KZKHRV.V1	0.0706	0.0000
MTMHRV.V5	0.3370	0.0000
MTMHRV.V4	0.2700	0.0000
MTTHRV.V5	0.3440	0.0000
MTTHRV.V4	0.2750	0.0000
MTTHRV.V3	0.2070	0.0000
WAWHRV.V2	0.3670	0.0000
WAWHRV.V3	0.3058	0.0000
WAAHRV.V2	0.1480	0.0000
WAAHRV.V1	0.0738	0.0000
QROHRV.V2	0.5100	0.0000
QROHRV.V1	0.4420	0.0348
QROHRV.V3	0.3711	0.0000
USIIHRV.V1	0.5973	0.0000
USIIHRV.V3	0.5347	0.0000
FADFHRV.V3	0.2710	0.0000
FADFHRV.V2	0.1450	0.0000
GYJYHRV.V6	0.4070	0.0679
GYJYHRV.V5	0.3400	0.0000

absolutely white pixels.) The second noise is the "frozen-frame" effect [4]. Random noise is present in every scene that we view, but it is not seen since the noise is not correlated from scene to scene. But when the scene is "frozen" (digitized), the random noise is clearly visible.

The first type of noise can often be removed by a simple threshold device that sets all pixels below a certain level to black, and all pixels above it to white. Thresholding was not used for two reasons. First, some of the noise would meet the threshold and would be printed as black as the characters. This could severely degrade the character shapes, causing the algorithm to fail. Second, and most important, leaving the noise unchanged was a good test of the algorithm's noise-handling ability. It works very well. It may work better without the noise; that should be checked in the future.

The noise from the frozen-frame effect can easily be removed by simple arithmetic averaging. The noise between any number of scenes is uncorrelated (the noise is assumed to be white), yet the picture should be highly correlated. The averaging process should rapidly reduce the noise, while not reducing the scene. Even though a simple average will remove this noise, it again was left it in to see if this algorithm would work with the noise; it did.

Bibliography

1. Nagy, G. "State of the Art in Pattern Recognition," Proceedings of the IEEE, Vol. 56, No. 5, pp. 836-859, May 1968.

2. Ascher, R. N., G. M. Koppelman, M. M. Miller, G. Nagy, and G. L. Shelton, Jr. "An Interactive System for Reading Unformatted Printed Text," IEEE Transactions on Computers, Vol. C-20, No. 12, pp. 1527-1543, Dec 1971.

3. Hoffman, J. and D. McCullough. "Segmentation Methods for Recognition of Machine Printed Characters," IBM Journal of Research and Development, Vol. 15, pp. 153-165, Mar 1971.

4. Kabrisky, Matthew. Lecture notes from EE 6.20 and EE 6.21. School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio. 1981.

5. Horev, Moshe. "Picture Correlation Model for Automatic Machine Recognition." Unpublished MS thesis, Wright-Patterson AFB, Ohio, Dec 1980. (AFIT/GE/EE/80D-25)

6. Kabrisky, Matthew. A Proposed Model for Visual Information Processing in the Human Brain, Urbana: University of Illinois Press, 1966.

7. Ginsburg, Arthur. "Psychological Correlates of a Model of the Human Visual System." MS thesis, Wright-Patterson AFB, Ohio, June 1971. (AFIT: GE/EE/71S-2) AD 731 197

8. Oppenheim, Alan V. and Ronald W. Schafer. Digital Signal Processing, Englewood Cliffs, NJ: Prentice-Hall, 1975.

9. Oppenheim, Alan V. and Jae S. Lim. "The Importance of Phase in Signals," Proceedings of the IEEE, Vol. 69, No. 5, pp. 529-541, May 1981.

10. Fredal, Dan and George C. Beasley, Jr. "An Analog Speech I/O Channel for the Nova 2 Computer Based on the S-100 Bus," MS thesis, Wright-Patterson AFB, Ohio, Mar 1981. (AFIT/GE/EE/81M-2) AD A103 398

11. Dorsey, Thomas and Darryl Stroup. "Scene Analysis: Application of Two-Dimensional Non-Linear Filtering for Target Enhancement." Unpublished MS thesis, Wright-Patterson AFB, Ohio, Dec 1981. (AFIT/GE/GEO/EE/81D-57)

12. Ankeney, Lawrence A.. "The Classification of Chinese Characters by Spatial Filtering Techniques." MS thesis, Wright-Patterson AFB, Ohio, Mar 1971. (AFIT: GE/EE/71-2) AD 722 852

Appendix

The source listing of each program listed in figure 3.2 (page 16) and described in chapter 3 is given in this Appendix. The listing is alphabetical, and listings for any Fortran subroutines that the main programs call are also included.

```
C       Subroutine CONVERT      By Lt. Simmons    19 July 81
C
C       This subroutine converts video pixels to lineprinter
C       pixels for the Printronix lineprinter.  This line-
C       printer requires that each printer dot in a row be
C       addressed by a bit of an octal number, six dots per
C       octal number.  A seventh bit must always be one.  The
C       eighth bit available in a byte is zero.  One peculiar-
C       ity is that the bits addressed are reversed in the
C       byte.  The right-hand dot in a row of six dots
C       becomes the sixth bit from the right going left in
C       the byte.  The first dot on the left in the row of
C       six dots becomes the first bit on the right in the
C       byte.  Note that two pixels are processed in each
C       call to CONVERT.
C
        SUBROUTINE CONVERT(IV,ILP)
        DIMENSION ILP(3)          ;Three words per pixel
        DO 1 II=1,3
    1   ILP(II)=100K              ;Zero the three words
        IVR=IV.AND.15             ;Strip off right pixel
        IVL=ISHFT(IV,-4).AND.15 ;Strip off left pixel
        IF(IVL.EQ.0)GO TO 0       ;Left pixel is zero
        GO TO(5,10,15,20,25,30,35,40,45,50,55,60,65,70,75)IVL
        STOP "Error in Subroutine CONVERT" ;Should not occur
C
C       Process left pixel
C
    0      ILP(1)=ILP(1)+007K           ;Left pixel is 0
           ILP(2)=ILP(2)+007K
           ILP(3)=ILP(3)+007K
           GO TO 75                 ;Jump to process right pixel
    5   ILP(1)=ILP(1)+007K             ;Left pixel is 1
        ILP(2)=ILP(2)+005K
        ILP(3)=ILP(3)+007K
        GO TO 75                    ;Jump to process right pixel
   10      ILP(1)=ILP(1)+006K        ;Left pixel is 2
           ILP(2)=ILP(2)+007K
           ILP(3)=ILP(3)+003K
           GO TO 75                 ;Jump to process right pixel
   15   ILP(1)=ILP(1)+007K           ;Left pixel is 3
        ILP(2)=ILP(2)+005K
        ILP(3)=ILP(3)+002K
        GO TO 75                    ;Jump to process right pixel
   20      ILP(1)=ILP(1)+005K        ;Left pixel is 4
           ILP(2)=ILP(2)+005K
           ILP(3)=ILP(3)+005K
           GO TO 75                 ;Jump to process right pixel
   25   ILP(1)=ILP(1)+005K           ;Left pixel is 5
        ILP(2)=ILP(2)+002K
        ILP(3)=ILP(3)+005K
        GO TO 75                    ;Jump to process right pixel
```

```
   30       ILP(1)=ILP(1)+002K          ;Left pixel is 6
            ILP(2)=ILP(2)+007K
            ILP(3)=ILP(3)+002K
            GO TO 75                 ;Jump to process right pixel
   35    ILP(1)=ILP(1)+002K          ;Left pixel is 7
         ILP(2)=ILP(2)+005K
         ILP(3)=ILP(3)+002K
         GO TO 75                    ;Jump to process right pixel
   40       ILP(1)=ILP(1)+005K          ;Left pixel is 8
            ILP(2)=ILP(2)+000K
            ILP(3)=ILP(3)+005K
            GO TO 75                 ;Jump to process right pixel
   45    ILP(1)=ILP(1)+004K          ;Left pixel is 9
         ILP(2)=ILP(2)+002K
         ILP(3)=ILP(3)+001K
         GO TO 75                    ;Jump to process right pixel
   50       ILP(1)=ILP(1)+002K          ;Left pixel is A
            ILP(2)=ILP(2)+000K
            ILP(3)=ILP(3)+005K
            GO TO 75                 ;Jump to process right pixel
   55    ILP(1)=ILP(1)+000K          ;Left pixel is B
         ILP(2)=ILP(2)+005K
         ILP(3)=ILP(3)+000K
         GO TO 75                    ;Jump to process right pixel
   60       ILP(1)=ILP(1)+004K          ;Left pixel is C
            ILP(2)=ILP(2)+000K
            ILP(3)=ILP(3)+001K
            GO TO 75                 ;Jump to process right pixel
   65    ILP(1)=ILP(1)+000K          ;Left pixel is D
         ILP(2)=ILP(2)+002K
         ILP(3)=ILP(3)+000K
         GO TO 75                    ;Jump to process right pixel
   70       ILP(1)=ILP(1)+004K          ;Left pixel is E
            ILP(2)=ILP(2)+000K
            ILP(3)=ILP(3)+000K       ;Continue for more processing
C
C        Process right pixel
C
   75    IF(IVR.EQ.15)RETURN            ;Right pixel is F
         IF(IVR.EQ.14)GO TO 80          ;Right pixel is E
         IF(IVR.EQ.13)GO TO 81          ;Right pixel is D
         IF(IVR.EQ.12)GO TO 82          ;Right pixel is C
         IF(IVR.EQ.11)GO TO 83          ;Right pixel is B
         IF(IVR.EQ.10)GO TO 84          ;Right pixel is A
         IF(IVR.EQ.9)GO TO 85           ;Right pixel is 9
         IF(IVR.EQ.8)GO TO 86           ;Right pixel is 8
         IF(IVR.EQ.7)GO TO 87           ;Right pixel is 7
         IF(IVR.EQ.6)GO TO 88           ;Right pixel is 6
         IF(IVR.EQ.5)GO TO 89           ;Right pixel is 5
         IF(IVR.EQ.4)GO TO 90           ;Right pixel is 4
         IF(IVR.EQ.3)GO TO 91           ;Right pixel is 3
         IF(IVR.EQ.2)GO TO 92           ;Right pixel is 2
```

```
      IF(IVR.EQ.1)GO TO 93              ;Right pixel is 1
C
         ILP(1)=ILP(1)+070K             ;Right pixel is 0
         ILP(2)=ILP(2)+070K
         ILP(3)=ILP(3)+070K
         RETURN
 80    ILP(1)=ILP(1)+040K              ;Right pixel is E
       ILP(2)=ILP(2)+000K
       ILP(3)=ILP(3)+000K
       RETURN
 81       ILP(1)=ILP(1)+000K           ;Right pixel is D
          ILP(2)=ILP(2)+020K
          ILP(3)=ILP(3)+000K
          RETURN
 82    ILP(1)=ILP(1)+040K              ;Right pixel is C
       ILP(2)=ILP(2)+000K
       ILP(3)=ILP(3)+010K
       RETURN
 83       ILP(1)=ILP(1)+000K           ;Right pixel is B
          ILP(2)=ILP(2)+050K
          ILP(3)=ILP(3)+000K
          RETURN
 84    ILP(1)=ILP(1)+020K              ;Right pixel is A
       ILP(2)=ILP(2)+000K
       ILP(3)=ILP(3)+050K
       RETURN
 85       ILP(1)=ILP(1)+040K           ;Right pixel is 9
          ILP(2)=ILP(2)+020K
          ILP(3)=ILP(3)+010K
          RETURN
 86    ILP(1)=ILP(1)+050K              ;Right pixel is 8
       ILP(2)=ILP(2)+000K
       ILP(3)=ILP(3)+050K
       RETURN
 87       ILP(1)=ILP(1)+020K           ;Right pixel is 7
          ILP(2)=ILP(2)+050K
          ILP(3)=ILP(3)+020K
          RETURN
 88    ILP(1)=ILP(1)+020K              ;Right pixel is 6
       ILP(2)=ILP(2)+070K
       ILP(3)=ILP(3)+020K
       RETURN
 89       ILP(1)=ILP(1)+050K           ;Right pixel is 5
          ILP(2)=ILP(2)+020K
          ILP(3)=ILP(3)+050K
          RETURN
 90    ILP(1)=ILP(1)+050K              ;Right pixel is 4
       ILP(2)=ILP(2)+050K
       ILP(3)=ILP(3)+050K
       RETURN
 91       ILP(1)=ILP(1)+070K           ;Right pixel is 3
          ILP(2)=ILP(2)+050K
```

```
            ILP(3)=ILP(3)+020K
            RETURN
92      ILP(1)=ILP(1)+060K              ;Right pixel is 2
        ILP(2)=ILP(2)+070K
        ILP(3)=ILP(3)+030K
        RETURN
93          ILP(1)=ILP(1)+070K          ;Right pixel is 1
            ILP(2)=ILP(2)+050K
            ILP(3)=ILP(3)+070K
            RETURN
        END
```

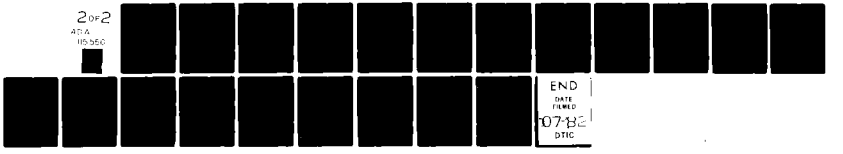
```
C       Program CREAD           Version 2
C
C       This program will calculate the magnitude and/or
C       phase of a complex 256 by 256 file.  The output mag-
C       nitude file is always called "RMAG" and the output
C       phase file is always called "ANG."  The input file
C       is specified in the command line.  The command line
C       format is:
C
C               CREAD/[M/P] Inputfile
C
C       where the global switch /M means create RMAG, and
C       the global switch /P means create ANG.  Both switches
C       can be be used.  At least one must be given.
C
        DIMENSION IO(1024),IO1(512),IO2(512),IFILE(7),MS(2)
        COMPLEX MA(256)
        COMMON MA,RMAG(256),ANG(256)
        EQUIVALENCE(MA(1),IO(1)),(IO1(1),RMAG(1)),(IO2(1),
     :  ANG(1))
C
C       Accomplish disk I/O tasks.
C
        CALL GROUND(I)
        IF(I.EQ.0)OPEN 0,"COM.CM"   ;Open ch. 0 to "COM.CM"
        IF(I.EQ.1)OPEN 0,"FCOM.CM"  ;Open ch. 0 to "FCOM.CM"
        CALL COMARG(0,I,MS,IER)  ;Read global switches
        IF(IER.NE.1)TYPE" COMARG1 error:",IER
        IF(MS(2).NE.0)STOP "Bad global switch."
        IF(MS(1).NE.1.AND.MS(1).NE.8.AND.MS(1).NE.9)STOP
     :  "Bad global switch."
        IF(MS(1).EQ.1)GO TO 1   ;Calculate phase only
        TYPE" RMAG will be created."
        DELETE "RMAG"
        CALL CFILW("RMAG",3,512,I)
        IF(I.NE.1)TYPE" RMAG CFILW error:",I
        IF(MS(1).EQ.8)GO TO 2   ;Calculate magnitude only
1       TYPE" ANG will be created."
        DELETE "ANG"
        CALL CFILW("ANG",3,512,I)
        IF(I.NE.1) TYPE" ANG CFILW error:",I
2       CALL COMARG(0,IFILE,I,IER)
        IF(IER.NE.1)TYPE" COMARG2 error:",IER
        OPEN 3,IFILE
        IF(MS(1).NE.1)OPEN 4,"RMAG"
        IF(MS(1).NE.8)OPEN 5,"ANG"
C
C       Calculate magnitude and/or phase.
C
        DO 5 I=0,255
        CALL RDBLK(3,(I*4),IO,4,IE)
        IF(IE.NE.1) TYPE" RDBLK error:",IE,I
```

```
      DO 4 J=1,256
      RMAG(J)=CABS(MA(J))
      IF(RMAG(J).EQ.0.0)GO TO 3
      X=REAL(MA(J))
      Y=AIMAG(MA(J))
      ANG(J)=ATAN2(Y,X)
      GO TO 4
3     ANG(J)=0.0
4     CONTINUE
    IF(MS(1).NE.1)CALL WRBLK(4,(I*2),IO1,2,IE)
    IF(MS(1).NE.1.AND.IE.NE.1) TYPE" WRBLK4=",IE
    IF(MS(1).NE.8)CALL WRBLK(5,(I*2),IO2,2,IE)
    IF(MS(1).NE.8.AND.IE.NE.1) TYPE " WRBLK5=",IE
5   CONTINUE
    CALL RESET
    STOP
    END
```

```
C       Program DISPLAY    By Lt. Simmons    14 Oct 1981
C       Version 7               Fortran IV
C
C       This program will convert video pixels to lineprinter
C       pixels, and will put the picture in a file or on the
C       Printronix 300 lineprinter.  This program prints
C       either the complete 256 by 256 pixel picture, or a
C       smaller picture that does not contain the noise
C       created by the video digitizer (the last five blocks
C       in the video file are noise).
C
        DIMENSION ILP(3,67),IFILE(7),IREC(64),ISAV(3)
        LOGICAL SHORT
C
C       Solid line, space, and line feed/plot-on characters
C
        IL=177777K                                  ;Solid line
        NC=40100K                                   ;Space
        LF=012K                                     ;Line feed
        LFPC=2412K                          ;Line feed/plot on
C
C       Set up parameters for complete picture display.
C
        SHORT=.FALSE.                       ;Short picture test
        N1=66           ;Top and bottom border length
        N2=256                  ;Number of lines displayed
        N3=1                    ;Location of left border
        N4=66                   ;Location of right border
        N5=67                   ;Location of line feed
        N6=1                    ;Length of lines displayed
C
C       Open the video file for input
C
        ACCEPT" What is the name of the input file? "
        READ(11,17)IFILE(1)     ;Read video file name
        CALL OPEN(1,IFILE,IER)          ;Open the video file
        IF(IER.NE.1)TYPE" Input open error:",IER
C
C       Ask for an output file, either the lineprinter or a
C       disk file, and open the disk file if necessary.
C
        ACCEPT" Do you want a disk file created (YES/NO)? "
  1     READ(11,19)N              ;Read one ASCII character
        IF(N.EQ.19968)GO TO 2     ;File was not selected
        IF(N.NE.22784)GO TO 20          ;Input error
        ACCEPT" A file was selected; what should its name be
     * (13 char max)? "
        READ(11,17)IFILE(1)       ;Read output file name
        CALL DFILW(IFILE,JER)       ;Make sure that the
        IF(JER.NE.1.AND.JER.NE.13)TYPE" Output delete error:"
     :  ,JER
        CALL CFILW(IFILE,2,KER)     ;file does not exist
```

```
      IF(KER.NE.1)TYPE" Output create error:",KER
      CALL OPEN(12,IFILE,LER) ;Open the output file
      IF(LER.NE.1)TYPE" Output file open error:",LER
      GO TO 3
   2  TYPE" The picture will only go to the lineprinter."
C
C     Choose complete picture or noiseless picture.
C
   3  CONTINUE
      ACCEPT" Do you want a complete picture (YES/NO)? "
   4  READ(11,19)N              ;Read one ASCII character
      IF(N.EQ.19968)GO TO 22          ;Response was "NO"
      IF(N.NE.22784)GO TO 21          ;Input error
      TYPE" A complete 256 by 256 pixel picture was chosen."
C
C     Put a border at the top of the picture.
C
   5  IF(SHORT)WRITE(12,18)           ;Double space first
      DO 7 I=1,3
         IF(SHORT)WRITE BINARY(12)NC          ;Space right
         IF(SHORT)WRITE BINARY(12)NC          ;twice
         DO 6 J=1,N1
   6     WRITE BINARY(12)IL           ;Print a line
   7     WRITE BINARY(12)LFPC         ;Terminate the line
C
C     Each line of the picture will have a border on each
C     end.  A DO-LOOP loops 233 (or 256 for whole frame)
C     times around the next three program parts
C
      DO 13 JA=1,N2
C
C     Put a border down the left hand side
C
      DO 8 K=1,3                           ;Put a left border
         IF(SHORT)ILP(K,1)=NC              ;two spaces in
         IF(SHORT)ILP(K,2)=NC              ;(~1/4" space)
   8     ILP(K,N3)=43500K                  ;for short picture
C
C     Put a border down the right hand side
C
      DO 9 L=1,3                           ;Insert border and
         ILP(L,N4)=40170K                  ;line feed after
   9     ILP(L,N5)=LFPC                    ;the picture rows
C
C     Convert the video picture pixels to $LPT pixels
C
      DO 10 M=1,64
  10     READ BINARY(1)IREC(M)          ;Read one video line
         DO 12 N=N6,64
            IWR=IREC(N).AND.377K        ;Right two pixels
            IWL=ISHFT(IREC(N),-8).AND.377K ;Left two pixels
            CALL CONVERT(IWL,ISAV)      ;Convert pixels
```

AD-A115 556 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOO--ETC F/G 9/2
MACHINE SEGMENTATION OF UNFORMATTED CHARACTERS.(U)
DEC 81 R A SIMMONS
UNCLASSIFIED AFIT/GE/EE/81D-54 NL



```
                N7=N+1
                IF(SHORT)N7=N
                DO 11 JB=1,3
   11              ILP(JB,N7)=ISHFT(ISAV(JB),8) ;Put in hi byte
                CALL CONVERT(IWR,ISAV)    ;Convert pixels
                DO 12 JC=1,3
   12              ILP(JC,N7)=ILP(JC,N7)+ISAV(JC) ;Put low byte
                DO 13 JE=1,3
                   DO 13 JD=1,N5
   13         WRITE BINARY(12)ILP(JE,JD)
C
C       Put a border and title at the bottom of the picture
C
        DO 15 JF=1,3
           IF(SHORT)WRITE BINARY(12)NC     ;Space right twice
           IF(SHORT)WRITE BINARY(12)NC     ;for short picture
           DO 14 JG=1,N1
   14      WRITE BINARY(12)IL              ;Print a line
   15      WRITE BINARY(12)LFPC    ;Terminate the line
        WRITE BINARY(12)LF      ;End with a line feed
        WRITE(12,16)                    ;Title picture
        CALL RESET                      ;Close all channels
        STOP
C
C       Format statements.
C
   16   FORMAT(' ',15X,'Signal Processing Laboratory, Air
     : Force Institute of Technology, Wright-Patterson AFB,
     : OH  45433<14>')
   17   FORMAT(S13)                     ;Filename format
   18   FORMAT('0')     ;Double space for short picture
   19   FORMAT(S1)              ;Query format
   20   ACCEPT" Input error.  Try again (YES/NO) > "
        GO TO 1
   21   ACCEPT" Input error.  Try again (YES/NO) > "
        GO TO 4
C
C       Set up parameters for shortened picture.
C
   22   TYPE" The shortened (noise removed) picture was
     : chosen."
        SHORT=.TRUE.                    ;Turn on short test
        N1=63                   ;Top and bottom border length
        N2=233                  ;Number of lines displayed
        N3=3                    ;Location of left border
        N4=65                   ;Location of left border
        N5=66                   ;Location of right border
        N6=4                    ;Length of lines displayed
        GO TO 5
        END
```

```
      SUBROUTINE HISTOGRAM(RMAX,RMIN,IFILE)
C
C     This subroutine will print out on the terminal
C     screen the number of values in 25 ranges across
C     the range RMIN to RMAX.
C
      DIMENSION RMAG(256),IFILE(7),NUMBER(25)
      COMMON IOM(512)    ;Integer array for mag. input
      EQUIVALENCE (RMAG(1),IOM(1))
      TYPE" Subroutine HISTOGRAM executing."
C
C     Zero the storage numbers.
C
      DO 1 I=1,25      ;25 storage locations
1     NUMBER(I)=0
      OPEN 1,IFILE     ;Open mag. file on ch. 1
      RINC=(RMAX-RMIN)*0.04            ;25 ranges
      DO 2 J=0,255              ;256 lines
         CALL RDBLK(1,(J*2),IOM,2,JER)
         IF(JER.NE.1)TYPE" RDBLK error:",JER
         DO 2 K=1,256           ;256 numbers in RMAG
            DO 2 L=1,25         ;25 ranges
               IF((RMAG(K).GE.(RMIN+(L-1)*RINC)).AND.
     :         (RMAG(K).LT.(RMIN+L*RINC)))NUMBER(L)=
     :         NUMBER(L)+1
               IF(NUMBER(L).EQ.32766)NUMBER(L)=32765
2              CONTINUE
      CLOSE 1          ;Close mag. file
      DO 3 M=1,12      ;Two lines at a time
         OUT1B=RMIN+((M-1)*RINC)       ;Bottom of 1st range
         OUT1T=RMIN+(M*RINC)           ;Top of 1st range
         OUT2B=RMIN+((M+11)*RINC)      ;Bottom of 2nd range
         OUT2T=RMIN+((M+12)*RINC)      ;Top of 2nd range
         N=M+12
3     WRITE(10,4)OUT1B,OUT1T,NUMBER(M),OUT2B,OUT2T,NUMBER(N)
4     FORMAT(' Range ',E11.4,' to ',E11.4,':',I5,' Range ',
     : E11.4,' to ',E11.4,':',I5)
      WRITE(10,5)OUT2T,RMAX,NUMBER(25)    ;Write 25th range
5     FORMAT(' ',39X,'Range ',E11.4,' to ',E11.4,':',I5)
      RETURN
      END
```

```
C       Program HRV     By Lt. Simmons          11 Sep 1981
C       Fortran 5
C       This program energy-normalizes a given magnitude
C       array, then combines it with a specified phase array
C       to produce a 256 by 256 array of complex numbers.
C          The command line format is:
C
C                HRV Magnitudefile Phasefile Outputfile
C
        INTEGER CFILE(7),PHFILE(7),C,PH
        DIMENSION MFILE(7),PHASE(1024)
        REAL MAG(1024),SUMSQ
        COMPLEX COUT(1024)
        COMMON IOM(2048),IOPH(2048),IOC(4096)
        EQUIVALENCE (IOM(1),MAG(1)),(IOPH(1),PHASE(1)),
     :  (COUT(1),IOC(1))
        M=1             ;Channel for magnitude file
        PH=2            ;Channel for phase file
        C=3             ;Channel for complex output file
C
C       Complete disk file tasks.
C
        CALL IOF(3,M1,MFILE,PHFILE,CFILE,I1,I2,I3,I4)
        OPEN M,MFILE            ;Open ch. 1 to mag. file
        OPEN PH,PHFILE          ;Open ch. 2 to phase file
        DELETE CFILE
        CALL CFILW(CFILE,3,1024,LER)    ;Create output file
        IF(LER.EQ.41)STOP "Insufficient contiguous blocks:
     :  Output file"
        IF(LER.NE.1)TYPE" Output file creation error:",LER
        OPEN C,CFILE            ;Open output file on ch. 3
    1   FORMAT(S13)
C
C       Read in magnitudes for energy normalization.
C
        SUMSQ=0.0       ;Variable for storing sum of squares
        Z=65536.                ;256 squared
        DO 2 I=0,504,8                  ;Loop 64 times
           CALL RDBLK(M,I,IOM,8,NER)    ;Read 8 blocks
           IF(NER.NE.1)TYPE" Mag. RDBLK error:",NER
           DO 2 J=1,1024
              SUMSQ=SUMSQ+(MAG(J)/Z)**2 ;Sum of squares
    2   CONTINUE
        SUMSQ=SQRT(SUMSQ)   ;Square root of sum of squares
        REWIND M                ;Rewind magnitude file
C
C       Read in magnitudes and phases and calculate complex
C       numbers.
C
        DO 4 K=0,504,8                  ;Loop 64 times again
           CALL RDBLK(M,K,IOM,8,IERR)   ;Read 8 mag. blocks
           IF(IERR.NE.1)TYPE" 2nd mag. RDBLK error:",IERR
```

```
      CALL RDBLK(PH,K,IOPH,8,JERR) ;Read 8 phase blks.
      IF(JERR.NE.1)TYPE" Phase RDBLK error:",JERR
      DO 3 L=1,1024
         ENM=MAG(L)/SUMSQ  ;Energy-normalize magnitudes
         X=ENM*COS(PHASE(L))       ;Real part
         Y=ENM*SIN(PHASE(L))       ;Imag part
3        COUT(L)=CMPLX(X,Y)   ;Create complex number
      CALL WRBLK(C,(K*2),IOC,16,KERR) ;Write complex #s
      IF(KERR.NE.1)TYPE" WRBLK error:",KERR
4     CONTINUE
      CALL RESET                      ;Close all channels
      STOP
      END
```

```
      SUBROUTINE IOF(N,MAIN,F1,F2,F3,MS,S1,S2,S3)
C
C     Written by Lt. Simmons          10 Sep 1981
C     Version 2
C
C     This FORTRAN 5 subroutine will read from the file
C     COM.CM (FCOM.CM in the foreground) the program name,
C     any global switches, and up to three local file
C     names and corresponding local switches.
C
C     Calling arguments:
C
C     N is the number of local files and switches to be
C     read from (F)COM.CM.  N must be 1, 2, or 3.
C
C     MAIN is an ASCII array for the main program file
C     name.
C
C     F1, F2, and F3 are the three ASCII arrays to return
C     the local file names.
C
C     MS is a two-word integer array that holds any global
C     switches.
C
C     S1, S2, and S3 are two-word integer arrays that
C     hold the local switches corresponding to F1 through
C     F3 respectively.
C
C
C     Dimension the arrays.
C
      DIMENSION MAIN(7),MS(2)
      INTEGER F1(7),F2(7),F3(7),S1(2),S2(2),S3(2)
C
C     Check the bounds on N.
C
      IF(N.LT.1.OR.N.GT.3)STOP "N out of bounds in IOF."
C
C     Process the data in (F)COM.CM
C
      CALL GROUND(I)  ;Find out which ground prog. is in
      IF(I.EQ.0)OPEN 0,"COM.CM" ;Open ch. 0 to COM.CM
      IF(I.EQ.1)OPEN 0,"FCOM.CM" ;Open ch. 0 to FCOM.CM
      CALL COMARG(0,MAIN,MS,IER) ;Read from (F)COM.CM
      IF(IER.NE.1)TYPE" COMARG error:",IER
      WRITE(10,1)MAIN(1)              ;Type program name
    1 FORMAT(' Program ',S13,'running.')
      CALL COMARG(0,F1,S1,JER)        ;Read from (F)COM.CM
      IF(JER.NE.1)TYPE" COMARG error (F1):",JER
      IF(N.EQ.1)GO TO 2               ;Test N
      CALL COMARG(0,F2,S2,KER)        ;Read from (F)COM.CM
      IF(KER.NE.1)TYPE" COMARG error (F2):",KER
```

```
        IF(N.EQ.2)GO TO 2                   ;Test N
        CALL COMARG(0,F3,S3,LER)            ;Read from (F)COM.CM
        IF(LER.NE.1)TYPE" COMARG error (F3):",LER
2       CLOSE 0
        RETURN
        END
```

```
      SUBROUTINE ITOC(PXLS,OUT)
C
C     This program will convert the video pixels passed
C     in PXLS to complex numbers (with imaginary parts
C     zero), and return them in OUT.
C
      INTEGER PXLS(1024)
      COMPLEX OUT(1024)
C
C     Note that PXLS has one dimension here, but is two-
C     dimensional in the calling program.  A single Do-loop
C     can access all values in a one-dimensional array
C     much more easily than a two-dimensional array.
C
      DO 1 I=1,1024
         BIMAG=0.0
    1    OUT(I)=CMPLX(FLOAT(PXLS(I)),BIMAG)  ;I -> C
      RETURN
      END
```

```
      SUBROUTINE LOGHIST(RMAXL,RMINL,IFILE)
C
C     This subroutine will print out on the terminal
C     screen the number of values in 25 ranges across
C     the range RMINL to RMAXL, using a logarithmic scale.
C
      DIMENSION RMAG(256),IFILE(7),NUMBER(25)
      COMMON IOM(512)   ;Integer array for mag. input
      EQUIVALENCE (RMAG(1),IOM(1))
      TYPE" Subroutine LOGHIST executing."
C
C     Zero the storage numbers.
C
      DO 1 I=1,25      ;25 storage locations
1     NUMBER(I)=0
      OPEN 1,IFILE     ;Open mag. file on ch. 1
      RINC=(RMAXL-RMINL)*0.04          ;25 ranges
      DO 2 J=0,255              ;256 lines
         CALL RDBLK(1,(J*2),IOM,2,JER)
         IF(JER.NE.1)TYPE" RDBLK error:",JER
         DO 2 K=1,256           ;256 numbers in RMAG
            DO 2 L=1,25         ;25 ranges
               IF((ALOG10(RMAG(K)).GE.(RMINL+(L-1)*RINC))
     :         .AND.(ALOG10(RMAG(K)).LT.(RMINL+L*RINC)))
     :         NUMBER(L)=NUMBER(L)+1
               IF(NUMBER(L).EQ.32766)NUMBER(L)=32765
2              CONTINUE
      CLOSE 1          ;Close mag. file
      DO 3 M=1,12      ;Two lines at a time
         OUT1B=RMINL+((M-1)*RINC)      ;Bottom of 1st range
         OUT1T=RMINL+(M*RINC)          ;Top of 1st range
         OUT2B=RMINL+((M+11)*RINC)     ;Bottom of 2nd range
         OUT2T=RMINL+((M+12)*RINC)     ;Top of 2nd range
         N=M+12
3     WRITE(10,4)OUT1B,OUT1T,NUMBER(M),OUT2B,OUT2T,NUMBER(N)
4     FORMAT(' Range ',E11.4,' to ',E11.4,':',I5,' Range ',
     : E11.4,' to ',E11.4,':',I5)
      WRITE(10,5)OUT2T,RMAXL,NUMBER(25)  ;Write 25th range
5     FORMAT(' ',39X,' Range ',E11.4,' to ',E11.4,':',I5)
      RETURN
      END
```

```
C       Program Pick    Version 1   Lt. Simmons    10 Sep 1981
C       This program will display a pixel magnitude only if
C       it falls between a user-specified range.  A histogram
C       (e.g. program PIX1) can list possible pixel magnitude
C       ranges.
C          The command line format is:
C
C                PICK Inputfile Outputfile
C
        INTEGER FILEI(7),FILEO(7),PIX(1024),VIDEO(256),SAVE(4)
        REAL MAG(1024)
        COMMON IOM(2048),MIO(2048)
        EQUIVALENCE (IOM(1),MAG(1))
C
C       Perform disk I/O tasks.
C
        CALL IOF(2,M1,FILEI,FILEO,I1,M2,I2,I3,I4)
        OPEN 1,FILEI                ;Open mag. file to ch. 1
        DELETE FILEO    ;Make sure that FILEO does not exist
        CALL CFILW(FILEO,3,64,IER)       ;on the disk
        IF(IER.EQ.41)STOP "Insufficient contiguous blocks:
     :  FILEO"
        IF(IER.NE.1)TYPE" CFILW error:",IER
        OPEN 2,FILEO
C
C       Request max. and min. levels for pixel inclusion.
C
        ACCEPT" Highest level to include? ",RMAX
        ACCEPT" Lowest level to include?  ",RMIN
C
C       Read all magnitudes and write magnitudes that are in
C       the specified range.
C
        DO 3 I=0,63
           CALL RDBLK(1,(I*8),IOM,8,JER)  ;Read 8 blocks
           IF(JER.NE.1)TYPE" RDBLK error:",JER
           DO 1 J=1,1024
              PIX(J)=15             ;Make all pixels white
              IF(MAG(J).LE.RMAX.AND.MAG(J).GE.RMIN)PIX(J)=
     :        IFIX(11.*(MAG(J)-RMIN)/(RMAX-RMIN)) ;Create pixel
1          CONTINUE                 ;Range of pixel creation loop
           DO 2 K=0,255
              DO 2 L=1,4
                 SAVE(L)=PIX((K*4)+L)   ;Save 4 pixels/loop
                 CALL REPACK(SAVE,VIDEO(K+1)) ;Repack 4 pixels
2             CONTINUE             ;Range of repacking loop
           CALL WRBLK(2,I,VIDEO,1,KER)  ;Write one block
           IF(KER.NE.1)TYPE" WRBLK error:",KER
3       CONTINUE                   ;Range of selection loop
        CALL RESET
        STOP
        END
```

```
C       Program PIX1 converts an input 256x256 real number
C       file into a VIDEO file.
C
C       Version 5                              2 Sep 1981
C
C       The command line format is:
C
C               PIX1[/H/L] Inputfile Outputfile
C
C       where the global switch /H causes a histogram to be
C       printed on the console, and adding the global /L
C       switch causes the histogram and the clip level
C       requests to be in base ten logarithm format.
C
        INTEGER OUTFILE(7)
        DIMENSION IO(2048),IO1(1024),IO2(256),INFILE(7),MS(2)
        COMMON RMAG(1024)
        EQUIVALENCE(RMAG(1),IO(1))
C
C       Disk I/O tasks.
C
        CALL IOF(2,M1,INFILE,OUTFILE,I1,I2,MS,I3,I4,I5,I6)
        IF(MS(1).NE.0.AND.MS(1).NE.256.AND.MS(1)
     :.NE.272..OR.MS(2).NE.0)STOP "Bad global switch."
        IF(MS(1).EQ.272)TYPE" Base ten logarithm histogram
     : option on."
        IF(MS(1).EQ.256)TYPE" Normal histogram option on."
        OPEN 3,INFILE
        DELETE OUTFILE
        CALL CFILW(OUTFILE,3,64,IE)
        IF(IE.NE.1)TYPE" Output CFILW:",IE
        OPEN 4,OUTFILE
C
C       Find min and max values of input file.
C
        RMIN=1.0E60
        RMAX=0.0
        DO 1 I=0,63
           CALL RDBLK(3,(8*I),IO,8,IE)
           IF(IE.NE.1) TYPE" RDBLK1 error:",IE
           DO 1 J=1,1024
              RMAX=AMAX1(RMAG(J),RMAX)
              RMIN=AMIN1(RMAG(J),RMIN)
    1         CONTINUE
        TYPE" Min:",RMIN,"   Max:",RMAX
        RMINL=ALOG10(RMIN)       ;Logarithm of RMIN
        RMAXL=ALOG10(RMAX)       ;Logarithm of RMAX
        TYPE" Min (log):",RMINL,"   Max (log):",RMAXL
        REWIND 3                 ;Rewind input file channel
C
C       Call histogram (if requested).
C
```

```
      IF(MS(1).EQ.256)CALL HISTOGRAM(RMAX,RMIN,INFILE)
      IF(MS(1).EQ.272)CALL LOGHIST(RMAXL,RMINL,INFILE) ;Log
C
C     Convert reals to gray scale integers and pack.
C
      IF(MS(1).EQ.256.OR.MS(1).EQ.0)ACCEPT" Clip gray scale
     : max?  ",RMAX1
      IF(MS(1).EQ.272)ACCEPT" Log clip gray scale max? ",
     :  RMAX1
      IF(MS(1).EQ.272)RMAX1=10.**RMAX1
      RMAX=AMIN1(RMAX,RMAX1)
      IF(MS(1).EQ.256.OR.MS(1).EQ.0)ACCEPT" Clip gray scale
     : min?  ",RMIN1
      IF(MS(1).EQ.272)ACCEPT" Log clip gray scale min? ",
     :  RMIN1
      IF(MS(1).EQ.272)RMIN1=10.**RMIN1
      RMIN=AMAX1(RMIN,RMIN1)
      DO 4 I=0,63
         CALL RDBLK(3,(I*8),IO,8,IE)
         IF(IE.NE.1) TYPE" RDBLK2 error:",IE
         DO 2 J=1,1024
            RMAG(J)=AMIN1(RMAG(J),RMAX)
            RMAG(J)=AMAX1(RMAG(J),RMIN)
            IO1(J)=IFIX(15.0*(RMAG(J)-RMIN)/(RMAX-RMIN))
2           CONTINUE
         KK=0
         DO 3 K=1,256
            IO2(K)=0
            DO 3 J=1,4
               KK=KK+1
               IO2(K)=ISHFT(IO2(K),4)
               IO2(K)=IO2(K)+IO1(KK)
3           CONTINUE
         CALL WRBLK(4,I,IO2,1,IE)
         IF(IE.NE.1) TYPE" WRBLK error:",IE
4        CONTINUE
      WRITE(10,5) OUTFILE(1)
      CALL RESET
      STOP
5     FORMAT(' ',S13,'created.')
      END
```

```
      SUBROUTINE REPACK(N,PIXELS,PXWD)
C
C     Written by Lt. Simmons          Version 2
C
C     This subroutine will repack four 4-bit integer pixels
C     into one 16-bit word for use by CHOPS.  Parameter N
C     allows more than one 4-bit to 1-word repacking
C     operation in each call to REPACK.
C
      INTEGER PIXELS(4,N),PXWD(N)
      DO 1 J=1,N                    ;Loop N times
      PXWD(J)=0
      DO 1 I=1,4
      PXWD(J)=ISHFT(PXWD(J),4)    ;Shift pixel left in word
    1 PXWD(J)=PIXELS(I,J)+PXWD(J) ;& add next pixel on rt.
      RETURN
      END
```

```
      SUBROUTINE UNPACK(N,PIXWORD,PIXELS)
C
C     Written by Lt. Simmons          Version 2
C
C     This subroutine will unpack four 4-bit integers from
C     a 16-bit integer word.  The pixels in a video file
C     have to be unpacked if each pixel is to be operated
C     on separately.
C
      INTEGER PIXWORD(N),PIXELS(4,N) ;Four pixels per word
      DO 1 I=1,N                     ;'N' allows higher-order
      DO 1 J=1,4                     ;arrays to be passed.
      PIXELS((5-J),I)=15.AND.PIXWORD(I) ;Pick off rt pixel
    1 PIXWORD(I)=ISHFT(PIXWORD(I),-4)   ;Shift word 4 bits
      RETURN              ;right to pick off next pixel.
      END
```

```
C       Program VIDEO    By Lt. Simmons    5 Aug 1981
C       Version 2.1
C
C       This program sets up parameters for a mode three
C       (video) call to the Cromemco Z-2D processor, via
C       the fortran subroutine CHANNEL.
C
        DIMENSION IPAR(2),IHOLD(7)
        INTEGER CROERR,NOVERR,FILE(7),PDCONT
        LOGICAL BTEST
        TYPE"NOTICE:  CHOPS must be running!<12>"
        ACCEPT"Input or output (IN-0/OUT-1)? ",IDIR
        IF(IDIR.NE.0.AND.IDIR.NE.1)GO TO 8 ;Error check
    1   J=0                                ;Dummy parameter
        K=3                                ;Mode 3 -> abort
C
C       Send an abort to the Cromemco to get its attention.
C
        CALL CHANNEL(J,J,K,J,J,"A",J,J,J,IE,IS)   ;Abort
        IDCNT=4                                   ;Set up
        IPCNT=1                                   ;parameters
        NTSK=3                                    ;for call to
        IM=2                                      ;CHANNEL
        ACCEPT"Enter time (SEC.):  ",IPAR(1) ;Display time
        ACCEPT"What is the name of the data file (13 Char
     : max)? "
        READ(11,8)FILE(1)          ;Video frame filename
        IF(FILE(1).NE.10752)GO TO 3       ;An "*" means to
        DO 2 I=1,7                       ;run with the last
    2   FILE(I)=IHOLD(I)                        ;file used
    3   IF(IDIR.EQ.1)GO TO 4
        CALL DELETE(FILE)    ;Delete file on input
    4   TYPE" Check monitor"
        CALL CHANNEL(NTSK,IDIR,IM,IPCNT,IDCNT,FILE,64,0,
     :  IPAR,IERR,ISYS) ;Call to CHANNEL
        IF(IERR.EQ.0.OR.(IERR.EQ.13323.AND.IDIR.EQ.0))
     :  GO TO 6     ;Jump if no errors
        IF(IERR.EQ.13323.OR.IERR.EQ.-24832.OR.IERR.EQ.
     :  -24064.OR.IERR
     :  .EQ.-22528)GO TO 5 ;Jump for specific error messages
        IF(BTEST(IERR,15))GO TO 9          ;Abnormal return
        GO TO 10                     ;Error without abort
    5   IF(IERR.EQ.-24832)TYPE"<7>ABORT--FILE DOES NOT EXIST"
        IF(IERR.EQ.13323.OR.IERR.EQ.-24064)TYPE"<7>ABORT--
     :FILE DOES NOT CONTAIN VIDEO"
        IF(IERR.EQ.-22528)TYPE"<7>ABORT--FILE CANNOT BE
     : CREATED"
    6   TYPE"<12>"
        DO 7 I=1,7                                ;Save filename
    7   IHOLD(I)=FILE(I)                          ;for next loop
        ACCEPT"What next (INPUT-0,OUTPUT-1,STOP-2)? ",IDIR
        IF(IDIR.EQ.0)GO TO 1
```

```
      IF(IDIR.EQ.1)GO TO 1
      IF(IDIR.EQ.2)STOP "Type VIDEO to rerun."
 8    FORMAT(S13)                ;Filename input format
      TYPE"<7>Input error; try again."
      GO TO 6
 9    TYPE"<7><12> ABORT INITIATED!!!<12>"    ;There are
10    CROERR=15.AND.IERR                      ;two error
      PDCONT=ISHFT(240.AND.IERR,-4)           ;codes in the
      NOVERR=ISHFT(-256.AND.IERR,-8)          ;variable
      TYPE" CROMEMCO ERROR RETURNED:",CROERR ;'IERR'
      TYPE" PARAMETER/DATA COUNT RETURNED:",PDCONT
      CALL BCLR(NOVERR,7)
      TYPE" NOVA ERROR RETURNED:",NOVERR ;Error
      TYPE" ERROR CODE RETURNED:",IERR   ;messages are
      TYPE" DATA COUNT:",IDCNT           ;printed for
      TYPE" PARAMETER COUNT:",IPCNT      ;user information
      TYPE" SYSERR RETURNED:",ISYS       ;and correction
      GO TO 6
      END
```

```
C***** Program VtoC ***********************************************
C                                                                  *
C       Version 5                                                  *
C                                                                  *
C       Written by Lt. Simmons          22 Sep 1981                *
C                                                                  *
C******************************************************************
C
C       This program will read in a video file (256 by 256
C       pixels) from a disk and will then write it out to a
C       disk file (either a contiguous or random file) as a
C       (256 by 256) complex array.  Note that the video file
C       is only 256 by 64, since four pixels are packed in
C       each 16-bit word in a video file.  These pixels have
C       to be "unpacked" before they can be written out as the
C       real parts of complex numbers.  The first 14 pixels
C       in each line can be blanked (made gray by setting to
C       8888 hex.).
C
C
C          The command line format is:
C
C               VTOC[/B] Inputfile Outputfile/[R or C]
C
C       where the global switch /B invokes blanking, and the
C       local switch (/R or /C) causes the Output file to be
C       either a random file or a contiguous file, respect-
C       ively.
C
        DIMENSION NAME(7),NOUT(7),MS(2),ISW(2)
        INTEGER VIDEO(256),PXLS(1024);256 words=1024 pixels
        COMPLEX OUT(1024)           ;1024 complex pixels
        LOGICAL Z
        COMMON IOUT(4096)
        EQUIVALENCE (OUT(1),IOUT(1))
        Z=.FALSE.
C
C       Open the video input and output files.
C
        CALL IOF(2,M,NAME,NOUT,I1,MS,I2,ISW,I3)
        OPEN 1,NAME                 ;Open ch. 1 to video file
        IF(ISW(1).EQ.8192)GO TO 2       ;Contiguous file
        IF(ISW(2).NE.16384)GO TO 1      ;Error--send message
        DELETE NOUT                 ;Ensure file does not exist
        CALL CFILW(NOUT,2,KER)          ;Create random file
        IF(KER.NE.1)TYPE" Random file creation error:",KER
        GO TO 3                         ;Continue
C
   1    TYPE" Bad local switch."
        CALL RESET
        STOP
C
```

```
    2    TYPE" Please wait while a contiguous file is created."
         DELETE NOUT              ;Ensure file does not exist
         CALL CFILW(NOUT,3,1024,LER)  ;Create contiguous file
         IF(LER.EQ.41)GO TO 10
         IF(LER.NE.1)TYPE" Contiguous file creation error:"
       : ,LER
C
    3    OPEN 2,NOUT              ;Open the file on ch. 2
C
         IF(MS(1).EQ.0.AND.MS(2).EQ.0)GO TO 5  ;No blanking
         IF(MS(1).NE.16384)GO TO 4       ;Error--send message
         TYPE" Gray blanking will occur."
         Z=.TRUE.                        ;Blank option on
         GO TO 5
    4    TYPE" Bad global switch."
         CALL RESET
         STOP
C
C        Process the change from video pixels to complex
C        numbers.
C
    5    DO 8 I=0,63              ;64 blocks per video file
            II=I*16
            CALL RDBLK(1,I,VIDEO,1,NER)  ;Read in 1 block
            IF(NER.NE.1)TYPE" RDBLK error:",NER
            IF((I.GE.59).AND.Z)CALL ZERO(VIDEO)  ;Blank
            IF(.NOT.Z)GO TO 7     ;Check for blanking
            DO 7 J=1,193,64
               DO 6 K=0,2         ;Blanking steps
    6             VIDEO(J+K)=VIDEO(J+K).OR.104210K ;Blank
                                                   ;12 bits
               VIDEO(J+3)=VIDEO(J+3).OR.10400K ;Blank 2 bits
    7          CONTINUE
            CALL UNPACK(256,VIDEO,PXLS)  ;Unpack four pixels
            CALL ITOC(PXLS,OUT)          ;Integer to complex
            CALL WRBLK(2,II,IOUT,16,IERR) ;Write complex #s
            IF(IERR.NE.1)TYPE" WRBLK error:",IERR
    8       CONTINUE
         CALL RESET
         STOP
C
C        Format statements.
C
    9    FORMAT(S1)                      ;Query format
C
C        Send error message for insufficient contiguous blocks.
C
   10    TYPE"<7><15>"
         TYPE" Sorry, but there are insufficient contiguous"
         TYPE" blocks for the output file to be created."
         TYPE" Would you like  to stop, or run with a"
         ACCEPT" random file (STOP/R)? "
```

```
11      READ(11,9)NSR
        IF(NSR.EQ.21248)STOP
        IF(NSR.NE.20992)GO TO 12
        TYPE" A random file will be created instead."
        CALL CFILW(NOUT,2,JERR)          ;Create random file
        IF(JERR.NE.1)TYPE" Random file creation error:",JERR
        GO TO 3
12      ACCEPT" Input error. Please try again (STOP/R) > "
        GO TO 11
        END
```

```
      SUBROUTINE ZERO(WORD)
C
CC    Version 2                                    22 Sep 1981
CCC
CCC   This subroutine gray blanks the word passed to it.
C
      INTEGER WORD(256)
      DO 1 I=1,256
    1 WORD(I)=104210K                ;Set word to 8888 (hex)
      RETURN
      END
```

Vita

Robin A. Simmons was born 7 October 1958 in Ionia, Michigan. He grew up in Saranac, Michigan and graduated valedictorian of his class from Saranac High School in 1976. He attended the University of Michigan at Ann Arbor, Michigan, as a four year AFROTC scholarship recipient, studying Electrical Engineering. Upon graduation in May 1980, he was commissioned a Second Lieutenant in the USAF as an AFROTC Distinguished Graduate. He was assigned to the School of Engineering, Air Force Institute of Technology.

Permanent Address: 5855 Sayles Road
Saranac, Michigan 48881

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFIT/GE/EE/81D-54	2. GOVT ACCESSION NO. AD-A115 556	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) MACHINE SEGMENTATION OF UNFORMATTED CHARACTERS		5. TYPE OF REPORT & PERIOD COVERED MS Thesis
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Robin A. Simmons 2Lt. USAF		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Air Force Institute of Technology (AFIT/EN) Wright-Patterson AFB, OH 45433		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS AFIT/ENG Wright-Patterson AFB, OH 45433		12. REPORT DATE December 1981
		13. NUMBER OF PAGES 114
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

15 APR 1982

18. SUPPLEMENTARY NOTES

Approved for public release IAW AFR 190-17

FREDRIC C. LYNCH, Major, USAF
Director of Public Affairs

E. Wolaver

Dean for Research and Professional Development
Air Force Institute of Technology (ATC)
Wright-Patterson AFB, OH 45433

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Character Segmentation
Fourier Transform
Optical Character Recognition
Pattern Recognition
Reading Machine

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This thesis presents a method of segmenting unformatted alpha-numeric characters. Reconstructing the magnitude of the Fourier transform of a template character with the phase of a string of unformatted characters containing the template character causes all characters that do not have the magnitude of the template to be attenuated in the visual domain. The template character will not be attenuated, since it has both proper magnitude and phase, and a peak detector can find the most probable location of the

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)


FILMED
7-8